1995

A study of multi-rigid-body dynamics: an application to robotic manipulators

Yijiang Fang
University of Wollongong

A STUDY OF MULTI-RIGID-BODY DYNAMICS

AN APPLICATION TO ROBOTIC MANIPULATORS

A thesis submitted in fulfilment of requirements

for the award of the degree

MASTER OF ENGINEERING (HONOURS)

from

UNIVERSITY OF WOLLONGONG, Australia.

by

YIJIANG FANG

(B.E., M.E.)

Department of Mechanical Engineering

January 1995

ACKNOWLEDGMENTS

I would like to express my deep appreciation to my supervisors, Assoc. Professor A. Basu and Dr. X. D. Fang, for their patient guidance, friendly advice, support and encouragement throughout the course of this research. They have been the best of advisors and teachers throughout my stay at Wollongong.

Thanks are also due to all the staff and the senior postgraduate students in the Dept. of Mechanical Engineering, who assisted in the completion of this work, specially to Mr. Des Jamieson for his invaluable help.

I wish to express my sincere appreciation to my wife, Jianfang Gu, and my son, Wei Fang, for their encouragement, support and unending patience. Finally, I am deeply grateful to my parents for their unfailing support and faith in me.

SUMMARY

In this dissertation, a general computer-oriented dynamical model for a serial robot manipulator is built based on D'Alembert's principle. The dynamical equations are derived on the basis of multi-rigid-body system according to structural characteristics of the serial robot manipulators. The equations are vectorial and completely recursive, and are suitable for a serial manipulator in general with revolute and/or prismatic joints.

Then, on the basis of the general dynamical equations obtained, an efficient recursive approach for the computer generation of a manipulator dynamical model is presented. It suits the direct or inverse problems of dynamics: The direct problem is that joint accelerations are computed from actuator forces; The inverse problem is that actuator forces are computed from joint displacements, velocities and accelerations. The method presented is a computer-oriented dynamical modelling method that forms and solves dynamics equations automatically. These dynamical equations obtained are succinct in form and compact in structure. Furthermore, the augmented body, defined as a fictitious body composed of link i and the mass of link i+1 through n attached at joint O_{i+1}, and composite system, defined as a body composed of link i through n, in the model for a serial robot arm are used. The recursive formulae corresponding to the augmented body and composite systems are set up, which make the dynamics computation simple and raise the computational efficiency. These formulae are very simple and straightforward, and can be used for solving the direct and inverse problem of dynamics.

On the basis of the dynamical model presented in this dissertation, we derived specific computational formulae and compiled general computer programs. In addition, a PUMA robot is taken as an example to illustrate the method on an IBM PC.

Keywords: Robotics: Dynamics modelling; Robot dynamics; Multi-rigid-body dynamics.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

LIST OF SYMBOLS

$\overline{\omega}_i$, $\dot{\overline{\omega}}_i$	The angular velocity and angular acceleration of link i.
$\overline{v}_i$, $\dot{\overline{v}}_i$	The linear velocity and acceleration at the mass centre of link i
$\ddot{\overline{\gamma}}_i$	The linear acceleration at the mass centre G_i of link i.
$\overline{F}_i$, $\overline{N}_i$	The total external force and moment with respect to mass centre exerted on link i.
$\bar{f}_i$, $\overline{n}_i$	The force and moment exerted on link i by link i-1.
τ_i	The torque exerted by actuator at joint i.
$\overline{p}_i^*$	The body vector of link i .
$H(q)$	The n×n symmetric inertia matrix.
$C(q,\dot{q})$	The n×n centrifugal and coriolis effects matrix
$G(q)$	The n×n gravity effects column matrix
$K(q)$	The 6×n Jacobian matrix.
k	The 6×1 external force and moment column matrix on link n.
q	The generalised coordinate
M_j	The mass of composite system j.
$\overline{C}_j$	The vector from the centre of mass of composite system j to the origin of the body coordinates of the link j-1.
$\underline{\underline{E}}_j$	The inertial tensor of the composite system j.
$\underline{\underline{J}}_i$	The inertial tensor of the link j.
m_j	The mass of link j.
$\overline{s}_j$	The vector from the origin of the body coordinates of the link j to the centre of mass of link j.
I	The 3×3 unit matrix.
${}^j\overline{n}_j$	The static moment of m_j with respect to joint j on the coordinate system j.

α_i,	The angle between the joint axes i and i+1.
a_i	The common normal distance between the joint axes i and i+1.
θ_i	The relative rotation angle of joint i.
d_i	The relative displacement of joint i.
$\overline{p}_j$	The vector from the origin of the inertia coordinate system to the origin of the coordinate system j.
W_j	The 3×3 coordinate transformation matrix from the inertia coordinate system to the coordinate system j.
$\tilde{p}$	The constant.
${}^{i}\overline{p}_i^*$	The vector from the origin of the coordinate system i-1 to the origin of the coordinate system i expressed on the coordinate system i.
A_{i+1}	The coordinate transformation matrix from the coordinate system i+1 to the coordinate system i.
T_i	The 4×4 coordinate transformation matrix from the coordinate system i to the inertia coordinate system.
ϕ_i	The 4×4 external force matrix acting on the link i.
Q_i	The inversion value of the torque (force) of the ith driver.
$R_{i-1,i}$	The 3×3 rotation (transformation) matrix from the coordinate system of link i to the coordinate system of link i-1.
$\overline{p}_{i-1,i}$	The 3×1 position vector of the origin of the coordinate system of link i in the Coordinate system of link i-l.
$\hat{u}$	The 3×3 skew symmetric tensor of vector $\overline{u}$.
$\tilde{W}$	The 3×1 vector corresponding to the 3×3 matrix W.
S_i	The type indicative symbol.
G_i	The mass centre of link i.
$\overline{\gamma}_i$	The position vector of the mass centre of link i relative to the origin O of base coordinate system.
$\overline{\pi}_i$	The virtual rotative angle.

$\overline{M}_i$	The total external force and moment with respect to mass centre exerted on link i.
a_{mk}	The elements of symmetric inertia matrix [A].
b_{mkj}	The elements of centrifugal and coriolis effects matrix [B].
E_m	The elements of gravity effects column matrix [E].
$\underline{\underline{\hat{d}}}_{mi}$	The skew symmetrical tensor of vector $\overline{d}_{mi}$.
M^i	The mass of augmented body i;
$\overline{u}^i$	The static moment of augmented body i with respect to point O_i.
$\underline{\underline{k}}^i$	The inertia tensor of augmented body i with respect to point O_i.
$\overline{U}^i$	The static moment of composite system i with respect to point O_i.
$\underline{\underline{K}}^i$	The inertia tensor of composite system i with respect to point O_i.
$\overline{x}^i$, $\overline{y}^i$, $\overline{z}^i$	The coordinates of mass centre of link i.
$\overline{X},\overline{Y},\overline{Z},I_{xx},I_{yy},I_{zz}$	The coordinates of mass centre and the main moments of inertia of the corresponding links.
HT	The step size.
N	The degree of freedom.

PUBLICATIONS during the studies for the degree of Master of Engineering (Honours)

1. Y. J. Fang, A. Basu, X. D. Fang and Z. Z. Wang, "A Efficient Recursive Approach For Computer Generation of Manipulator Dynamic Model". The International Journal of *Mathematical and Computer modelling*, Vol. 20, No. 9, pp. 89-96, 1994, Pergamon Press.

2. Y. J. Fang, A. Basu and X. D. Fang, "A New Manipulator Dynamics Modelling Method", The Proc. of *Third International Conference on Automation, Robotics and Computer Vision*, TP4.1, pp. 967-971, November 9-11 ., 1994, Singapore.

3. Y. J. Fang, A. Basu and X. D. Fang, "Dynamic Analysis of Robotic Manipulators with Flexible Links", Accepted subject to revision in 1994 for the International Journal of *Laboratory Robotics and Automation*..

4. X. D. Fang, Y. J. Fang and S. Hamidnia, "Computer Animation of 3-D Chip Formation in Oblique Machining", Paper submitted for review for Trans. ASME, Journal of Engineering for Industry.

CHAPTER 1

INTRODUCTION

This dissertation deals with the kinematics and dynamics of multi-rigid-body systems, in particular, manipulators. The aim is to develop an efficient approach for computer generation of manipulator dynamical model on basis of multi-rigid-body system dynamics.

In the following section some remarks are made concerning previous work in the fields of kinematics and dynamics of multi-rigid-body systems, in general, and manipulators, in particular.

1.1 Dynamics of Multi-Rigid-Body Systems

In recent years, there has been increasing interest in the development of equations of motion for multi-rigid-body systems--that is, systems containing many rigid bodies. There are two main reasons for this interest: First, many complicated mechanical systems and devices such as manipulators, robots and biosystems, can be effectively modelled by systems of rigid bodies; and second, it has just recently become possible, with the aid of high-speed digital computers, to obtain efficient numerical solutions of the governing dynamic equations. The emphasis of researchers working with multi-rigid-body systems has therefore been the formulation of equations of motion which can easily be developed into numerical algorithms for computer codes.

A multi-rigid-body system consists of a number of rigid bodies (referred to as links) connected in succession by kinematic pairs (referred to as joints). The kinematic

pairs allow relative motion between the two bodies they connect. One end of the system is generally fastened to a base reference while the other end, referred to as the terminal link, is free to move about in space.

From the view point of analytical mechanics, multi-rigid-body systems are extremely interesting and challenging multi-degree-of-freedom mechanisms. In comparison to the closed-loop single-degree-of-freedom mechanism the multi-rigid-body system, with computer programming, is a more versatile general purpose mechanical device, such as the manipulator. In the analysis of a multi-rigid-body system, two main problems are encountered. The first problem, referred to as the inverse kinematics or positioning problem, is to obtain a set of joint displacements which will position the terminal link at a given location with a specified orientation in space. The second problem, referred to as the inverse dynamical problem, is to obtain the joint forces and/or the joint torques which will produce specified joint positions, velocities, and accelerations.

The emergence of industrial manipulators has led to a renewed interest in the study of multiple-degree-of-freedom rigid-body motions. The need for the development of an extensive and rigorous theoretical basis, in this field, is felt more and more in engineering design both from the point of view of analysis as well as from that of synthesis.

1.1.1 Review of History

The dynamics of multi-rigid-bodies is a new science branch developed on the basis of the classical mechanics. The classical rigid mechanics represented by Euler and Lagrange formulations are in use for long time. Up to the middle of the twentieth century little, if any, effort was put on the dynamic analysis of systems of

interconnected bodies undergoing general translation and rotation - what in this work are called multi-rigid-body systems. Everything needed to do such analyses existed except two items: a motivation which could not be ignored or circumvented, and a key to unlock the heavy door of practicality.

By the end of the Eighteenth Century the basic laws of motion for both translation and rotation were known. Some problems involving single rigid bodies had been solved. A unified formalism, that of Lagrange, was available to treat systems of bodies.

By mid-Twentieth Century special techniques for handling linear vibrational problems had been well developed and were in general use, although the computational burden of numerical linear algebra limited their practical application to relatively small-scale systems. By that time there was considerable interest in nonlinear systems, but the mathematical difficulties of nonlinear mechanics focused much of the research on approximate literal or numerical methods and confined solutions largely to systems with one degree of freedom. Applications to engineering problems were not undertaken very often [1].

Systems of connected bodies undergoing very general motions had been encountered in various technological contexts. These include, as seen previously, mechanisms, gyroscopes in gimbals, ships, aircraft, road vehicles, rail vehicles and biomechanics. Only the spacecraft was still missing, but this was a fatal omission.

A variety of interconnections between the bodies of "real" multi-rigid-body systems was to be seen, including rotation about a common point, rolling, sliding.

Each of these motions can have several facets. Most physical devices permit relative rotation only about a common axis fixed in each of two contiguous bodies. However, in cases where gimbals do not have to be treated as separate bodies of the

system, relative rotation can be modelled as a motion with two or three degrees of freedom taking place about "mathematical gimbals" or about no gimbals at all. It already has been mentioned that rolling under extreme loading can differ significantly from rolling under light loading. Sliding can occur in a straight line or along a curved path. Finally, all of these situations can exist within the same interconnection, separately from one another or coupled (as a screw couples rotation and translation) [48].

Thus, by mid-Twentieth Century there seemed to be ample technological motivation to study the dynamic behaviour of multi-rigid-body systems. Why were such studies not done?

Sometimes the need for dynamic analysis was blunted by the permanent importance of kinematics, as in mechanisms. Sometimes it was possible to get around the multi-rigid-body aspects by artifice, as when the relative rotation of contiguous bodies remained so small that their inertia tensors could be added and treated as constant. But the dominant reasons for the failure of widespread multi- body dynamic analysis to develop beyond the case of small vibrations of linear systems were the facts that: firstly, essential nonlinearities usually made literal analysis impossible; secondly, practical limitations on time and effort made numerical analysis infeasible.

1.1.2 Multi-Rigid-Body System Dynamics

In the 1960 decade the situation changed completely. The first and most important impetus was the rise of the electronic digital computer, which finally broke down the massive computational barriers. The other was the need for detailed analysis of practical multi-rigid-body systems in two areas of application: the design of spacecraft and of high speed mechanisms. The multi-rigid-body nature of some spacecraft could

not be ignored, and there was a tremendous premium on doing the dynamic analysis before the system was constructed and flown. One simply could not afford to design by "cut and try". For mechanism designers the need for a mathematical analysis was not so strong. Nevertheless, an increasing need was felt in the early 1960s to use the computer as an aid to abbreviate the design process for high speed mechanisms where dynamic effects become important.

Investigations of multi-rigid-body models of spacecraft on the one hand and high speed mechanisms on the other led to the evolution of two schools of multi-rigid-body system dynamics, which did not take much notice of one another until the 1980s. In both areas, the treatment of specific cases started in the early 1960s. Special purpose computer programs were written as needed, limited largely to the case at hand. This was a reasonable approach for the day. It might appear that the separate programming for many cases would be inefficient, but the programs can be modularized so that all of them might use certain packages of coding in common. A significant advantage is that special purpose programs are easy to learn and use.

Toward 1965 attention turned to the possibility of constructing "general purpose" multi-rigid-body computer programs. With such a program one can use a single program to simulate the motion of a very broad class of multi-rigid-body systems. General purpose multi-rigid-body programs are capable of generating and integrating equations of motion, based only on input data describing the way the bodies are interconnected, the mechanical and geometric properties of the bodies and the interactions between them, together with the state of the system at an initial time. Thus the user of the program is free of the time-consuming and error-prone task of deriving complicated equations of motion for each system to be considered. As usual, of course, one has to pay for such an advantage: it takes considerable time and effort to master these general purpose programs. However, experience has demonstrated that

their advantages more than compensate for this disadvantage. Once mastered, the general purpose programs become powerful and time-saving tools.

The first developments of such general purpose computer programs were for limited classes of multi-rigid-body systems [49]. It is at this point that the two schools began to diverge. Mechanisms often are composed of closed circuits of links moving in a plane or in parallel planes. Thus, early developments in that field - mostly based on Lagrangian dynamical equations - were oriented toward such systems. The capability of dealing with systems moving in three dimensions came later. In contrast, the topological structure of a multi-rigid-body spacecraft usually is a tree [49]. (Cutting the interaction between any two connected bodies causes the system to fall into two parts.) The investigation of tree configurations of bodies is considerably simpler than those with closed mechanical circuits, because one does not have to consider any kinematical consistency conditions on the motions of the bodies forming the circuits. However, another complication is typical for spacecraft: they move in three-dimensional space. Consequently, spatial motions of tree-configured systems were investigated in the earliest publications dealing with spacecraft dynamics. Most of these approaches were based on the Newton/Euler equations and principles of dynamics [49].

During the 1970 decade, early developments were generalised by enlarging the class of systems encompassed by computer programs. The ability to treat closed circuits and three-dimensional motion was added. About 1980, communication between the two schools of general purpose program users began to develop [2--5]. The computational difficulties of simulating multi-rigid-body systems have been largely surmounted, but the problem of modelling the external and internal forces on the bodies still remains. Current progress is controlled by this factor, as in such examples as modelling wheel-rail interactions for rail vehicles, control of magnetically

levitated vehicles, control laws for robots and treatment of small clearances in the joints of high speed mechanisms.

Besides the problems of dealing with three-dimensional motion and closed mechanical circuits, a third important topic is the elasticity of the system's bodies. The earliest efforts in both the special case and the general purpose categories were based on rigid, or possibly gyrostatic, bodies. Subsequently, both approaches were generalised to allow elasticity, subject to certain restrictions, in the individual bodies.

1.1.3 Examples of Applications

Spacecraft and mechanisms form multi-rigid-body systems whose external environment is predictable and relatively simple. The same is largely true for the interactions between the bodies. Good dynamic models can be constructed, without too much difficulty, which give satisfactory results in the light of physical experience. Once multi-rigid-body programs had demonstrated their good performance in spacecraft and mechanism dynamics, they began to be used in other areas of technology. The main problems in these new fields of application appear when modelling the interactions.

A typical multi-rigid-body spacecraft dynamical problem of the late 1960s was the deployment of solar arrays [6]. The multi-rigid-body model was used to settle questions about the safety of the deployment procedure. Models for the hinge joint interactions and external actions upon the bodies are simple in this case.

Interactions are also simple to model in a typical early investigation of a closed circuit mechanism, such as an automobile hood connected to the frame of the car and a railway vehicle. The multi-rigid-body model was used in [7] to investigate the influence of design parameters on maximum speed and curving performance.

Although the model has more than forty degrees of freedom it is relatively simple except for the computation of the interaction between the wheels and the rails. Forces and torques across the interface depend on material parameters, but also on the size and shape of the contact patch and the relative velocities of the wheel and rail materials in the patch, the latter motion resulting from elasto-plastic deformation. The calculations of the contact forces are extremely complicated and, to some degree, are still a research problem area. Unfortunately, the wheel-rail interactions are not the only difficult modelling problem in simulation of vehicle dynamics. Hydraulic or pneumatic actuators may be used to improve performance of railway or highway vehicles (including passenger cars, trucks and busses) by introducing active control of the suspension. One concludes from [8] that the modelling of the corresponding forces and torques is one of the major problems in vehicle dynamic simulation.

Similar situations are encountered in the control of magnetically levitated ground vehicles, as described in [9]. One of the multi-rigid-body models was used to investigate this vehicle. It is relatively simple and straightforward as a dynamic model, but the derivation of models for the magnetic force between guideway and undercarriage (see [10], for example) and the laws for active feedback control require considerable effort.

The same is true for the example of a manipulator. Attaching sensors to its hand enables this machine to perform complicated manipulative tasks without external mechanical assistance. The principal problem in this case is the design and analysis of closed loop controls for the forces and torques in the individual joints of the robot arms. (See [11].) This problem is especially difficult when there are redundant degrees of mechanical freedom in the manipulator or robot.

The examples above represent some of the areas in which multi-rigid-body formalisms are being used. Others appear in connection with such problems as the

dynamics of aircraft landing gears investigated with the programs described in [12], and in the swinging and jackknifing of tractor semitrailers as in [13]. Agricultural and military vehicles have been studied as well. Still another example is the parachute model which has been used to study the swinging motion of the attached load. A field studied extensively in biomechanics is the simulation of the motion of crash victims in cars and airplanes [14].

1.1.4 Goals of Multi-Rigid-Body System Investigation

Before looking further into the nature of multi-rigid-body system analysis it is necessary to be explicit about what is meant by the term "dynamic simulation". Put a little differently, what is the reason for the general-purpose computer program to which we have referred?

To do a dynamic simulation of a system means to construct a mathematical model of the system and, by examining the behaviour of the model under various conditions, determine how a real system would behave if and when it is built. Except for some very simple systems, before the advent of electronic computers the only model within the investigator's grasp was a replica of the real thing. Some systems lend themselves to physical modelling but others do not, because of the impracticability of constructing a physical model of certain conditions (for example, free space without gravity), the cost of constructing a physical model, or the difficulty of freeing the physical model from disturbances or actions other than those that are to be modelled intentionally. Some people feel that physical models usually are worthless whereas others swear by them. However, this difference of opinion is not as important a question as it once was. For any system having reasonably great complexity the model nowadays is almost certain to be a mathematical model.

Mathematical modelling of a dynamic system means the formulation of mathematical relationships which describe all the features of the system that are considered essential - which is to say, all those which are believed to have a significant effect on the behaviour of the physical system being modelled. For example, to evaluate the ride comfort experienced by passengers of a railway vehicle it often suffices to take account of just the vertical motion. The corresponding dynamic model will lead to a set of equations describing the planar motion in the vertical plane of symmetry containing the vehicle's longitudinal axis. On the other hand, when investigating the maximum safe operating speed one also must take lateral motion into account, so the same vehicle must be analysed using a different, more sophisticated, mathematical model.

Ideally, one would like to solve these models analytically, even if only approximately. If this can be done the dependence of system behaviour on various design parameters can be seen explicitly, in literal terms. However, literal solutions are almost always impossible for multi-rigid-body systems. Therefore, when mathematical modelling is discussed in the context of multi-rigid-body systems it is a foregone conclusion that the model will be used mainly to get numerical solutions on a digital computer. Accordingly, the formulation of the model should be done with the needs of computation primarily in mind. It is quite possible to exercise a mathematical model using an analogue or hybrid computer, but for the purposes of this work we assume that the numerical solution is digital. In this case one speak of "digital simulation".

One function of a computer simulation is to permit an investigator to explore the behaviour of a system before it is built, to discover the effects of changes in system configuration and parameter values on its behaviour. Design, even of the simplest systems, seldom can be done by algorithm. In practice it almost always is done by successive analysis. A tentative design is conceived and its behaviour is found by

analysis - literal analysis, if possible, but otherwise by computer simulation. The design then is modified and the process is repeated until the designer is satisfied with his creation. Iterative design often can be viewed as a three-phase process. The first phase concerns qualitative feasibility. In light of the function for which the system is being designed, the designer has to establish whether his general concept is a feasible one. Phase two is performance evaluation. Usually there are quantitative requirements on system performance, so quantitative changes in behaviour when design parameters and initial conditions are changed become important. Phase three is optimisation. A final choice must be made, subject to all applicable constraints, for all parameters under the designer's control so that the system has the "best" possible performance.

Synthesis is basic to all technological applications, and computer simulation is a key to each part of the design process. Thus a general-purpose multi-rigid-body computer program should be regarded as a fundamental design tool.

The dynamical behaviour usually remains important throughout the system's life, so the value of computer simulation does not suddenly disappear after the system is built. Parameter values can change, external environment can change, failures can occur, operation can be deliberately modified, data analysis can create needs for the prediction of motion. The general-purpose simulation remains not just a useful tool, but an essential one.

1.2 Dynamics of Manipulators

In the recent years, the scientists all over the world have increasingly paid great attention to the research of robotics. The research of robotics requires the knowledge of multi-disciplines. It is a developing frontier science area leaping over several disciplines. Because of the developments of the mechanisms, multi-rigid-body dynamics, computer technology and modern control theory the technology of robot has reached a new step at present.

The robot is an important research area of modern science. "Where are the robots?" Several years ago there were forecasts that predicted a huge growth in robot numbers, yet these predictions have not come true. Why not? Perhaps robots are too expensive, too complicated or unsuited to the tasks we imagined they could perform. We have shown that robots can walk and run, play table-tennis, juggle drive trucks at high speed and fly in space. Yet the automotive industry is the only one where robots proliferate.

Along with the vigorous development of the science and technology, modern industry has set some still higher demands on the robot. Those demands make robots develop toward high speed, heavy load and more accuracy. At the same time, the dynamic real-time control has become essential to the development of the manipulator. However, the dynamic characteristics of manipulators are highly nonlinear and coupling. To control these systems, it is necessary to search for a computer-oriented algorithm that fast forms and solves dynamical equations automatically. This makes necessary the establishment of a satisfactory manipulator dynamic model.

The manipulator consists of a group of rigid-bodies. It is a typical multi-rigid-body system. In the opinion of mechanisms, manipulator is a kind of space mechanism system with multi-degrees of freedom. Manipulator structures are divided into two main classes: simple and complex chain robot arms. The simple chain robot

arm is a series of links and joints forming one kinematic chain structure. This type of robot arms was studied by many authors, Kahn, Bejczy, Raibert, Luh, Hollerbach, Paul, Megahed, the author himself and many others [15-26]. The complex chain type is divided into two main groups: one contains closed loops in its structure, and the other has no loops (Fig.1.1). The first group may have one or more end-effectors, while the second one has a tree shape. The dynamics of complex structure robot arms have been studied less than the simple chain type.

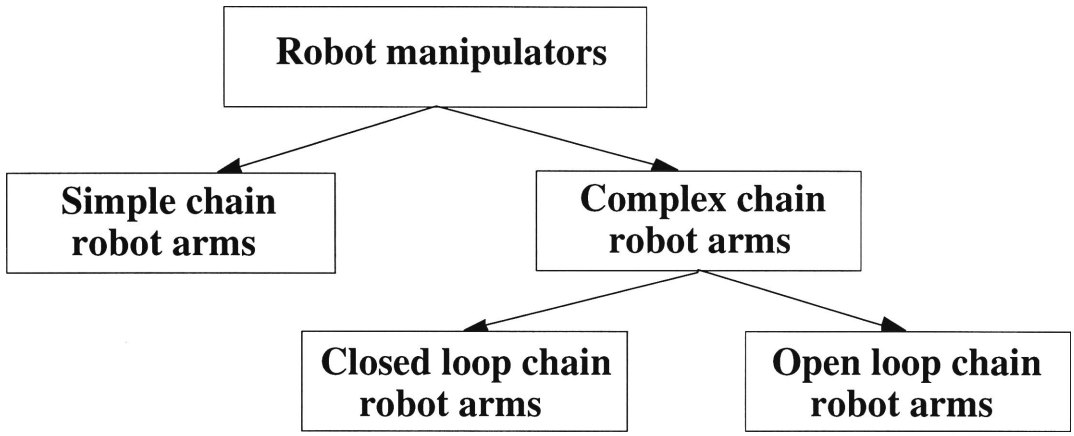

Fig 1.1 The structure of robot manipulators

Most manipulators are simple chain structure with joints between links that are generally the lower-pair with one degree of freedom. Owing to these characteristics, it is a kind of concrete and relatively simple multi-rigid-body system. Therefore, if we study it directly by the method of multi-rigid-body dynamics, we will make the simple problem more complicated. So, we should derive succinct and effective algorithms according to these characteristics of manipulators when we study the modelling method of manipulator dynamics.

The dynamics of any mechanical system can be studied using either Newton's equilibrium equations, D'Alembert's principle, or Lagrange equations of motion. Manipulator dynamics and mechanisms deal mainly with the mathematical formulation of the equations of manipulator motion.

Dynamical modelling of manipulators is important for both control system design and implementation. Manipulator dynamics can usually be modelled by a set of governing equations that describe the link motion characteristics mathematically. These governing dynamic equations are needed for manipulator control. The dynamics of manipulators can be categorised into two parts: the inverse dynamics and the forward dynamics. Many modern control schemes for manipulators require the inverse dynamics that determine the actuator forces for the prescribed joint displacements, velocities and accelerations. The forward dynamics are required while the motion simulation of manipulators is performed. The object is to solve the joint accelerations from dynamical equations when the actuator forces are given as input values. The joint velocities and displacements can be obtained by integrating the joint accelerations, and are then used to calculate the bias vector. The research of the forward dynamical focuses on the formulation of the inertia matrix and the bias vector.

For manipulator control, the complexity of computation in inverse dynamics is crucial. This computational complexity can be affected by several factors, namely: (1) the ways of formulating the dynamic equations; (2) the computational schemes; (3) the geometric structure of the robot, etc. Since the computational complexity is always the bottleneck of manipulator real-time control, any of the aforementioned factors that yield efficient computation of inverse dynamics is preferable.

With regard to the ways of formulating manipulator dynamical equations, some commonly used methods are Newton-Euler method, Lagrange method, Kane's method, and Generalised D'Alembert's method. Although the joint torques/force obtained from different methods for the same manipulator ought to be identical, the computational complexity in solving the equations depends fully on the forms of the equations. Sometimes, the form of equations derived by one method can be transformed to the form of the other. For example, in a recursive formulation of the dynamics of an open-chain manipulator, Silver showed that the form of the Lagrange equations of motion

can be equivalent to the Newton-Euler equations of motion by using the same representation of rotational dynamics [27]. However, as applied to a manipulator, different methods in general result in different forms of equations of motion.

The computational scheme is also an important factor to the reduction of inverse dynamical calculations. Some existing schemes such as the recursive Newton-Euler method [18, 28] and the recursive Lagrange method [30] for open-chain manipulators have been proven to be much more computationally efficient than the nonrecursive methods.

In deriving the dynamical equations, the following goals were maintained:

(a) the algorithm had to be capable of being implemented on a low-cost microcomputer which was computationally fast, to meet the servo rate demands of manipulators;

(b) the model had to be accurate, with all the dynamic terms explicitly represented, making no simplifications; this guarantees that an accurate positioning control will be achieved;

(c) the formulation had to be simple and straightforward to understand; this makes it easy to implement in parallel processing and multi-tasking, so reducing the order of computations.

The dynamic equations of a robot arm are highly non-linear functions. To control such a system, it is necessary to compute the forces and torques needed to drive its joints accurately and frequently at adequate sampling frequency of 60 Hz or more. In other side, to simulate the system also needs fast computation speed. So, how to raise the computation speed is the key to both inverse and forward dynamics. The computational organisation has a great effect on the necessary computation time and the required memory size. The organisation of the necessary computations is very important to minimise the execution time. Researchers on robotics have researched

this problem from two sides: (1) Developing new algorithms from machines to raise the computation speed; (2) Taking research from computation tools and means to raise the computation speed [50].

1.2.1 The Approach by Mechanics

In the last two decades, many succinct and efficient algorithms are presented according to the topological structure characteristics of manipulators. They are divided into three main classes: numerical method, symbolic method and numeric-symbolic method.

1) Numerical method

The so called numerical method is that all variables are treated as real numbers during the process that the mathematical model is formed and solved in computer. Each variable is determined by the corresponding numerical value. According to the form of equations built, the numerical method can be divided into two classes: recursive and non-recursive. In 1985 M.W.Walker [29] took a comparative study to the two approaches. On the number of computation, the non-recursive approach has a lot of repeat computation and the recursive approach can avoid these repeat computations. So, the number of computation of the recursive approach is obviously much less than that of the non-recursive approach. On the viewpoint of machine it is one of the important ways of realising real-time control to derive a succinct, efficient, complete recursive algorithm of manipulator dynamics. References [18, 28-31] have done much significance research work on this area. In these publications, the recursive Newton-Euler method [18] proposed by J.Y.S.Luh, M.W.Walker and R. Paul in 1980 and the state analysis method [31] developed from the former are much more computational efficient than others.

2). Symbolic method

The so called symbolic method is that the relative variables of manipulators are expressed by a group of symbols and run in symbolic form in the computer, then the equations expressed by the symbols are derived directly.

Several symbolic programs have been developed in the past decade, for example, ARM [32], NEWEUL [33], MESA VERDA [34], SO/EXACT [35], AUTOLEV [36] and PIODYM [37]. But it is a complicated problem to form the symbolic model by computer. Generally, a dynamical method of manipulators with n degrees of freedom has 18n variables [48]. So, high performance computers are required.

The use of symbolic operations is computationally quite time consuming, and the algorithm based on it cannot be executed on a mini- or micro-computer that does not have the necessary capability; for a manipulator with six degrees of freedom, the total number of operations for deriving its linearized dynamic models by means of the symbolic operations of Neuman and Murray [32] exceeds 150,000 multiplications and additions.

3). Numeric-symbolic method

Owing to the big memory space needed by symbolic method the variables of manipulators are divided into numerical variables and symbolic variables. This makes the operation of a part of variables transferred to numerical calculation. Therefore, the aim to minify the necessary memory can be reached. Furthermore, the constants are put together in the process of computation. So, the variables only consist of generalised coordinates and the symbolic expresses only are the functions of generalised coordinates. This is the so call numeric-symbolic method.

The applicability of this method is not much better because it dependents on too many on the kinetics and dynamics parameters of manipulators.

1.2.2. The Approach by Computational Tools and Means

On the area the parallel algorithm is main research topic. In order to shorten the calculation time for solving the inverse dynamical problem and satisfying the requirement of real-time control, Two or more microprocessors are used to carry out parallelly the calculation to raise the computation speed and to shorten the computation time.

In this method the variables of dynamical equations are grouped in order to be computed parallelly by multi-microprocessors. Obviously, this is an efficient method for increasing computation efficiency and saving running time. This method is under development.

1.3 Scope of the Dissertation

This dissertation deals with the kinematics and dynamics of multi-rigid systems to apply to manipulators. The aim is to develop an efficient approach for computer generation of manipulator dynamical model on the basis of multi-rigid-body system dynamics.

To reduce the computational time to the maximum in the dynamical real-time calculation of manipulators is the main aim of dynamical research of manipulators. To convert a part of calculation of equations to off-time calculation and to make the other calculation completely recursive is an important method for achieving this aim. In this dissertation, a new complete recursive dynamical modelling method to manipulators is presented which can be used to shorten the calculating time for solving the inverse dynamical problem and satisfy the requirement of real-time control. This is a computer-oriented method that can fast form and solve the equations automatically. The equations are derived via D'Alembert's principle on the basis of multi-rigid-bodies

dynamics according to the structural characteristics of the serial robot arm. The equations are vectorial and complete recursive in form and compact in the structure and the recurrence formula obtained is very simple and direct. Furthermore, we used the augmented body and composite system for the serial manipulator in this dynamical model, which makes the dynamical calculation simple and raises the computational efficiency. It suits the positive or inverse problems of dynamics.

1.3.1 The problem formulation

The problem formulation of this research can be stated as: How to build a computer-oriented dynamical modelling method with a high computational efficiency compared to other rigid-body modelling technique.

In order to solve this problem, the following tasks are considered:

- To derive a general dynamical model in order to suit for all kind of serial manipulators with revolute and/or prismatic joints.
- To avoid the appearance of partial differential equations in the dynamical model in order to suit computers. By doing so, the equations do not have to be derived by hand.
- To introduce the concepts of the augmented bodies and composite systems on multi-rigid-body dynamics to serial manipulators according to the structural characters of serial manipulators in order to get a succinct and compact equation.
- To adopt a complete recursive approach to build the model of dynamics in order to get a higher computational efficiency.
- The dynamical model has to suit the direct and inverse dynamical problems in order that a general program can be developed by this model, and the equations can be formed and solved by computer automatically.

1.3.2 The outline of dissertation

The dissertation is organised into six chapters. The outline of each chapter is given as follows.

Chapter 1: This chapter contains an introduction on the history and the applications of multi-rigid-body systems and the goals of this research. The outline of the dissertation is also included.

Chapter 2: In this chapter, the computational efficiency of algorithms is surveyed.

Chapter 3: In this chapter, a new general form of equations for the dynamical behaviour of manipulators is presented according to structural characteristics of a manipulator on the basis of multi-rigid-body system dynamics. The equations are vectorial and complete recursive and can be used for a serial manipulator with revolute and/or prismatic joints. It suits to the direct or inverse problems of dynamics.

Chapter 4: In this chapter, we present a numerical efficient recursive approach for the generation of a manipulator dynamic model on a computer. It forms and solves dynamical equations automatically. Furthermore, we use the augmented body and composite system for the serial robot arm in the dynamical model. The recursive formulae corresponding to the augmented body and composite systems are set up, which makes the dynamical computation simple and raises the computational efficiency. The recurrence formulae obtained are simple and direct, and can be used to shorten the calculation time for solving the inverse dynamical problem.

Chapter 5: In this chapter, the computational formulae are derived and the algorithms of the direct dynamics and the inverse dynamics based on the modelling technique presented in chapter 4. Finally, a case study for the PUMA 560 robot is presented.

Chapter 6: In this chapter, the concluding remarks are presented.

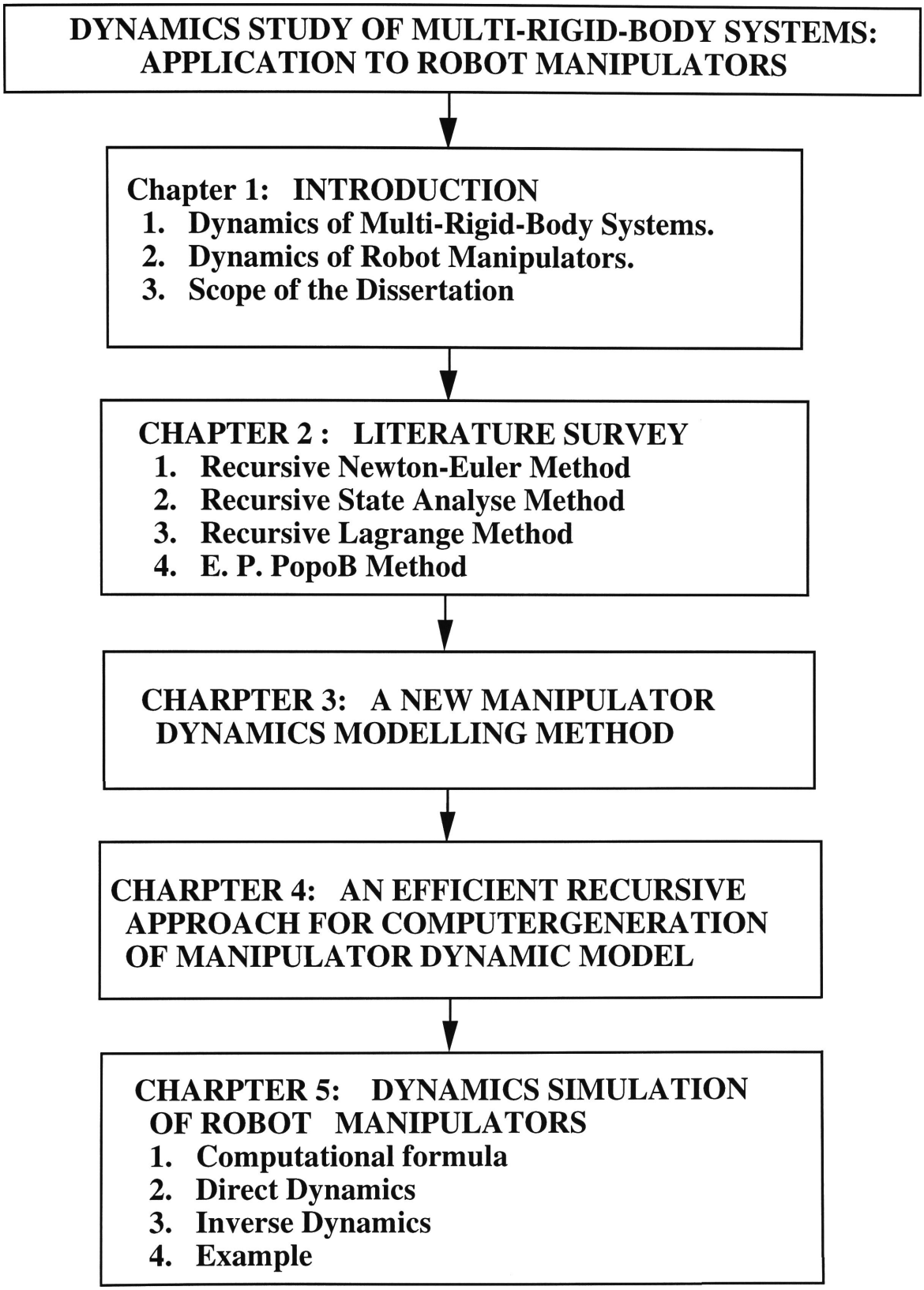

Fig. 1-2 Structure of the dissertation

CHAPTER 2

LITERATURE SURVEY

1. Introduction

Since the first robot came into the world, the countries all over the world have launched out into the research and development of robot in succession. For a short duration of one or two decades, the widespread and rapid development of robot technology has been achieved. In order to realise the real-time control of robot, a large amount of research work has been done on the dynamics of robot and many dynamical algorithms of robot have been presented. These algorithms are based on different mechanics principles, and have different structural forms and mathematical expressions. So, each of them possesses the respective characteristics.

The formulation of computationally-effective and efficient dynamics has been an active area of research for the last two decades, and several methods have been developed. The Lagrange-Euler approaches by Ucker [51], Kahn [52], Bejczy [53], and Paul [54] are very well structured and systematically represented, but suffer from very high computational complexity. This is mainly due to the nonrecursive properties of the computation. The "recursive lagrangian" method of Hollerbach [30] gives good computational results, but destroys the structure of the equations. The Newton-Euler approach of Orin et al. [55], Luh et al. [18] and Walker et al. [31] has the most efficient computational formulation, but untidy recursive equations. Other approaches include the tabulation techniques of Raibert and Horn [56], Kane's dynamical equations (Kane and Levinson [57]), Appel's method (Vukobratovic and Potkonjak [58]), and the generalised D'Alembert method of Lee and Lee [59]. The use of parallel processing

(e.g. Lathrop [60], Lee and Chang [61], Binder and Herzog [62]) and symbolic computation (e.g. Murray and Neuman [63], Cheng et al. [64]) has also been pursued by researchers

In this chapter, serials representative and high computational efficient algorithms are surveyed, and the main content and characteristics are introduced summarily.

In the following presentation, in order to simplify the presentation, only the situation that the relative displacement is revolute angle is considered. We define the follows:

Links and joints are numbered from 1 to n starting from the base (link 0); the ith joint situated between links i and i-1; the axis of the ith joint is defined by vector $\overline{Z}_i$; the O_i is a fixed point on this axis; the base coordinates ($\overline{X}_1, \overline{Y}_1, \overline{Z}_1$) is attached to link i with origin O_1; the coordinates ($\overline{X}_{i+1}, \overline{Y}_{i+1}, \overline{Z}_{i+1}$) is attached to link i with origin O_{i+1}.

2.2 Recursive Newton-Euler Method [18]

J.Y.S. Luh, M.W. Walker and R.P.C. Paul divided a manipulator into single rigid-body by cutting the joints of a manipulator by means of the Newton-Euler method, and then study these single rigid-body. Therefore, a group of succinct recursive formulations was derived. This method utilises the characteristics of manipulators, and builds a set of efficient algorithms to form the dynamical equations directly, and raises the computational efficiency.

From the principle of mass centre movement and the principle of momentum moment:

$$\begin{cases} \overline{F}_i = m_i \ddot{\overline{\gamma}}_i \\ \overline{N}_i = \underline{\underline{J}}_i \cdot \dot{\overline{\omega}}_i + \dot{\overline{\omega}}_i \times (\underline{\underline{J}}_i \cdot \overline{\omega}_i) \end{cases} \tag{2.1}$$

From the balance relative with respect to the joint:

$$\begin{cases} \bar{F}_i = \bar{f}_i - \bar{f}_{i+1} \\ \bar{N}_i = \bar{n}_i - \bar{n}_{i+1} + (\bar{p}_{i-1} - \bar{\gamma}_i) \times \bar{F}_i - \bar{p}_i^* \times \bar{f}_{i+1} \end{cases} \tag{2.2}$$

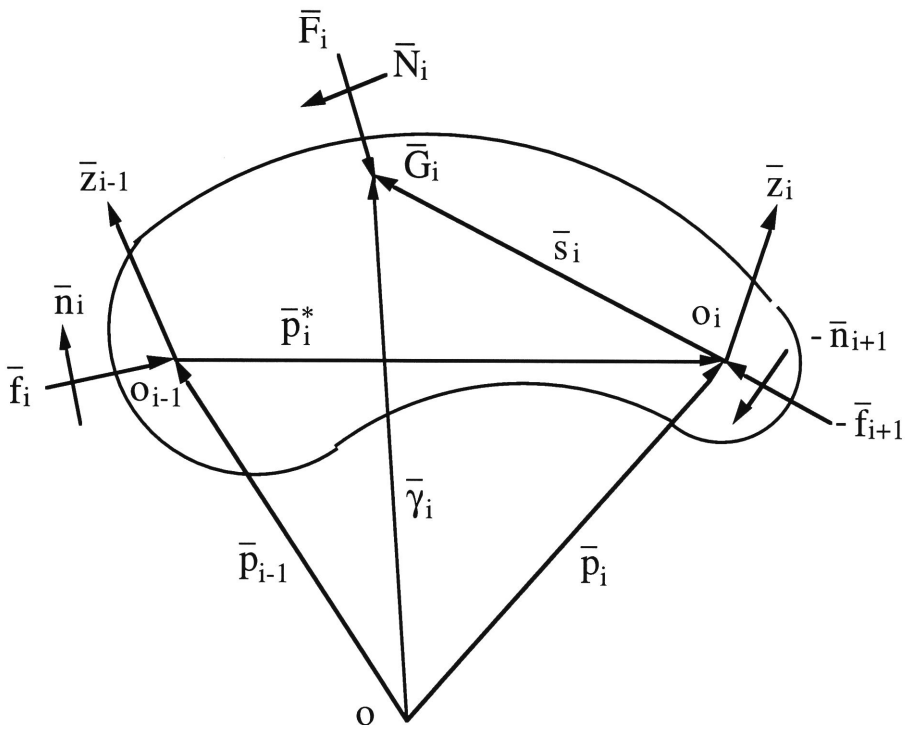

Fig. 2.1 The relation between the coordinate systems

Note $(\bar{\gamma}_i - \bar{p}_{i-1}) = (\bar{p}_i^* + \bar{s}_i)$, the dynamical recursive formula can be obtained:

$$\begin{cases} \bar{f}_i = \bar{F}_i + \bar{f}_{i+1} \\ \bar{n}_i = \bar{n}_{i+1} + \bar{p}_i^* \times \bar{f}_{i+1} + (\bar{p}_i^* + \bar{s}_i) \times \bar{F}_i + \bar{N}_i \\ \tau_i = \bar{z}_{i-1} \cdot \bar{n}_i \qquad\qquad (i = n, \ldots, 1) \end{cases} \tag{2.3}$$

According to the principle of relative movement, the kinematics recursive formulations can be derived:

$$\begin{cases} \overline{\omega}_i = \overline{\omega}_{i-1} + \bar{z}_{i-1}\dot{q}_i \\ \dot{\overline{\omega}}_i = \dot{\overline{\omega}}_{i-1} + \bar{z}_{i-1}\ddot{q}_i + \overline{\omega}_{i-1} \times \bar{z}_{i-1}\dot{q}_i \\ \ddot{\overline{p}}_i = \dot{\overline{\omega}}_i \times \overline{p}_i^* + \overline{\omega}_i \times (\overline{\omega}_i \times \overline{p}_i^*) + \ddot{\overline{p}}_{i-1} \\ \ddot{\overline{\gamma}}_i = \dot{\overline{\omega}}_i \times \bar{s}_i + \overline{\omega}_i \times (\overline{\omega}_i \times \bar{s}_i) + \ddot{\overline{p}}_i \\ \qquad\qquad (n = 1, \ \ldots, \ n) \end{cases} \tag{2.4}$$

Above two groups of the recursive formula are called inner recursive formula of dynamics and outer recursive formula of kinematics.

Here :

$\overline{\omega}_i$, $\dot{\overline{\omega}}_i$ --- The angular velocity and angular acceleration of link i.

$\ddot{\overline{\gamma}}_i$ --- The linear acceleration at the mass centre G_i of link i.

$\overline{F}_i$, $\overline{N}_i$ --- The total external force and moment with respect to mass centre exerted on link i.

$\bar{f}_i$, $\overline{n}_i$ --- The force and moment exerted on link i by link i-1.

τ_i --- The torque exerted by actuator at joint i.

$\overline{p}_i^*$, $\bar{s}_i$ --- The body vectors of link i (see Fig.2.1).

Owing to that the kinematics equation is built on each independent rigid-body in this method, every link is in independent equilibrium when the motion is given. The calculation is done under the body coordinates. It is not necessary to do all calculation under the inertia coordinates. So, the geometry and inertia parameters of each link are constants in the body coordinates themselves. Therefore, the number of calculation is reduced, and the calculation efficiency is raised. In addition, the recursive relative of this method is simple and direct, and easy to program. It is a simpler and more efficient algorithm of manipulators, but only suitable for solving inverse problem of dynamics. So, many researchers have done a lot of work on the basis of this method and much improvement and development has been made. Comparatively speaking, the M.W.Walker and D.E.Orin's research result [31] is better than others.

2.3. Recursive State-Analyse Method [31]

This method is derived by M.W.Walker and D.E.Orin on the basis of the recursive Newton-Euler method. In this method solving the complicated direct problem of dynamics is converted into solving the simpler inverse problem of dynamics under specified state conditions. It is suitable for direct and inverse problems of dynamics.

The equation is expressed as following:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + K(q)^T k = \tau \tag{2.5}$$

Here:

$H(q)$ --- The n×n symmetric inertia matrix.

$C(q,\dot{q})$ --- The n×n centrifugal and coriolis effects matrix

$G(q)$ --- The n×n gravity effects column matrix

$K(q)$ --- The 6×n Jacobian matrix.

k --- The 6×1 external force and moment column matrix on link n.

τ --- The n×1 drive force (moment) column matrix.

Let $B = C(q,\dot{q})\dot{q} + G(q) + K(q)^T k$

The equation becomes:

$$H(q)\ddot{q} = \tau - B \tag{2.6}$$

Note that the $H(q)$ only is the function of generalised coordinate q. Once the position of system is given, the H can be determined. At the same time, the all terms of B are independent of the generalised coordinate acceleration. So, the two specified state conditions can be defined as following.

State 1: Assuming that the system is under the state that the generalised coordinate accelerations $\ddot{q}_i$ are all zero:

$$\ddot{q}_i = 0 \qquad (i = 1, \ldots, n)$$

then $B = \tau$

The calculation of n×1 column matrix B can be converted into calculating the drive force τ under specified state $\ddot{q}_i = 0$.

State 2: Considering that the system is under a space, and let

$$\begin{cases} \dot{q}_i = 0 \quad (i = 1, \ldots, n); \\ \ddot{q}_j = 1, \ \ddot{q}_i = 0 \quad (i \neq j, \ i = 1, \ldots, n). \end{cases}$$

then $B = 0$ in equation (2.6), and the equation (2.6) become as:

$$H(q)\ddot{q} = \tau \tag{2.7}$$

The equation (2.7) can be expressed in the following form:

$$\begin{bmatrix} H_{11} & \cdots & H_{1j} & \cdots & H_{1n} \\ \cdot & & & & \cdot \\ \cdot & & H_{ij} & & \cdot \\ \cdot & & & & \cdot \\ H_{n1} & \cdots & H_{nj} & \cdots & H_{nn} \end{bmatrix} \begin{bmatrix} 0 \\ \cdot \\ 1 \\ \cdot \\ 0 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \cdot \\ \cdot \\ \cdot \\ \tau_n \end{bmatrix}$$

By expanding it, one can obtain:

$$\begin{bmatrix} H_{1j} & \cdots & H_{nj} \end{bmatrix}^T = \begin{bmatrix} \tau_1 & \cdots & \tau_n \end{bmatrix}_j^T$$

These expressions mean that the elements of jth column of H equal the drive force τ under the state 2. Carrying through this equation by j from n to 1 yields the all elements of H. Finally, solving the direct problem of dynamics is converted into solving the inverse problem of dynamics under specified state.

Computing τ under state 1 (in fact the B is computed):

$$\begin{cases} \overline{\omega}_i = \overline{\omega}_{i-1} + \bar{z}_{i-1}\dot{q}_i \\ \dot{\overline{\omega}}_i = \dot{\overline{\omega}}_{i-1} + \overline{\omega}_{i-1} \times \bar{z}_{i-1}\dot{q}_i \\ \ddot{\bar{p}}_i = \dot{\overline{\omega}}_i \times \bar{p}_i^* + \overline{\omega}_i \times (\overline{\omega}_i \times \bar{p}_i^*) + \ddot{\bar{p}}_{i-1} \\ \ddot{\bar{\gamma}}_i = \dot{\overline{\omega}}_i \times \bar{s}_i + \overline{\omega}_i \times (\overline{\omega}_i \times \bar{s}_i) + \ddot{\bar{p}}_i \\ \qquad\qquad (n = 1, \ \ldots, \ n) \end{cases} \tag{2.8}$$

$$\begin{cases} \overline{F}_i = m_i \ddot{\bar{\gamma}}_i \\ \overline{N}_i = \underline{\underline{J}}_i \cdot \dot{\overline{\omega}}_i + \overline{\omega}_i \times (\underline{\underline{J}}_i \cdot \overline{\omega}_i) \\ \bar{f}_i = \overline{F}_i + \bar{f}_{i+1} \\ \overline{n}_i = \overline{n}_{i+1} + \bar{p}_i^* \times \bar{f}_{i+1} + (\bar{p}_i^* + \bar{s}_i) \times \overline{F}_i + \overline{N}_i \\ \tau_i = \bar{z}_{i-1} \cdot \overline{n}_i \qquad\qquad (i = n, \ \ldots, \ 1) \end{cases} \tag{2.9}$$

Computing τ under state 2 (in fact the H is computed):

$$\begin{cases} \overline{F}_j = \bar{z}_{j-1} \times (M_j \overline{C}_j) \\ \overline{N}_i = \underline{\underline{E}}_j \cdot \bar{z}_{j-1} \qquad (j = 1. \ \ldots, \ n) \end{cases} \tag{2.10}$$

$$\begin{cases} \bar{f}_i = \bar{f}_{i+1} \\ \overline{n}_i = \overline{n}_{i+1} + \bar{p}_i^* \times \bar{f}_{i+1} \qquad (i = j - 1, \ \ldots, \ 1) \\ \bar{f}_j = \overline{F}_j \\ \overline{n}_j = \overline{N}_j + \overline{C}_j \times \overline{F}_j \end{cases} \tag{2.11}$$

$$\tau_i = \bar{z}_{i-1} \cdot \overline{n}_i$$

Where (the system of n-j+1 links from link j to link n is defined as composite system):

M_j --- The mass of composite system j.

$\overline{C}_j$ --- The vector from the centre of mass of composite system j to the origin of the body coordinates of the link j-1.

$\underline{\underline{E}}_j$ --- The inertial tensor of the composite system j.

$\underline{\underline{J}}_i$ --- The inertial tensor of the link j.

The recursive formulations for computing the parameters of composite system are given by:

$$\begin{cases} M_j = M_{j+1} + m_j \\ \overline{C}_j = \dfrac{1}{M_j}\left\{m_j(\bar{s}_j + \bar{p}_j^*) + M_{j+1}(\overline{C}_{j+1} + \bar{p}_j^*)\right\} \\ \underline{\underline{E}}_j = \underline{\underline{E}}_{j+1} + M_{j+1}[(\overline{C}_{j+1} + \bar{p}_j^* - \overline{C}_j)I - (\overline{C}_{j+1} + \bar{p}_j^* - \overline{C}_j) \\ \qquad \cdot (\overline{C}_{j+1} + \bar{p}_j^* - \overline{C}_j)^T] + \underline{\underline{J}}_i + m_j[(\bar{s}_j + \bar{p}_j^* - \overline{C}_j)I \\ \qquad -(\bar{s}_j + \bar{p}_j^* - \overline{C}_j)\cdot(\bar{s}_j + \bar{p}_j^* - \overline{C}_j)^T] \\ \qquad\qquad (j = n, \ \ldots, \ 1) \end{cases} \tag{2.12}$$

The initial conditions are given as follows:

$$\begin{cases} M_n = m_n \\ \overline{C}_n = \bar{s}_n + \bar{p}_n^* \\ \underline{\underline{E}}_n = \underline{\underline{J}}_n \end{cases} \tag{2.13}$$

where

m_j --- The mass of link j.

$\bar{s}_j$ --- The vector from the origin of the body coordinates of the link j to the centre of mass of link j.

I --- 3×3 unit matrix.

Walker and Orin utilised the concept of composite system in the algorithm of solving the inverse problem of dynamics, which improves the recursive Newton-Euler method and raises the computational efficiency.

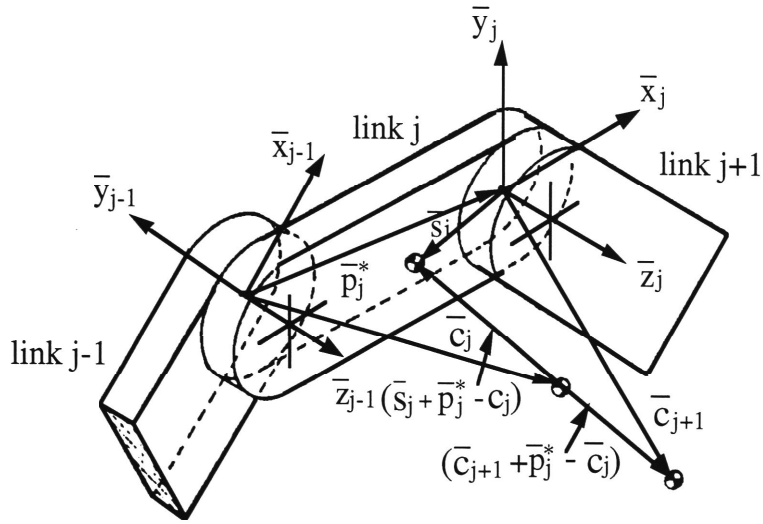

Fig. 2.2 Illustration of composite centre of mass of links j through N

2.4. Recursive Lagrange Method [30]

J.M.Hollerbach built a set of the systematic recursive algorithm on the basis of the Lagrange method, and obtained a group of recursive formulas expressed by 3×3 matrices and vectors.

The Lagrange function can be expressed as following:

$$L = \frac{1}{2}\sum_{j=1}^{n} tr(m_j \dot{\bar{p}}_j \dot{\bar{p}}_j^T + 2\dot{W}_j {}^j\bar{n}_j \dot{\bar{p}}_j^T + \dot{W}_j \underline{\underline{J}}_j \dot{W}_j^T) - \tilde{p} + \sum_{j=1}^{n} m_j \bar{g}^T W_j {}^j\bar{\gamma}_j \tag{2.14}$$

where

m_j --- The mass of link j.

${}^j\bar{n}_j$ --- The static moment of m_j with respect to joint j on the coordinate system j.

$\underline{\underline{J}}_j$ --- The inertia tensor of link j.

$\overline{p}_j$ --- The vector from the origin of the inertia coordinate system to the origin of the coordinate system j.

W_j --- The 3×3 coordinate transformation matrix from the inertia coordinate system to the coordinate system j

$\tilde{p}$ --- The constant.

Substituting the Lagrange function into the Lagrange equation:

$$\tau_i = \frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} \qquad (i = 1, \ldots, n) \tag{2.15}$$

The following equation can be derived:

$$\tau_i = tr(\frac{\partial W_i}{\partial q_i} D_i) - g^T \frac{\partial W_i}{\partial q_i} C_i \tag{2.16}$$

The recursive relatives can be expressed as follows:

$$\begin{cases} W_i = W_{i-1} A_i \\ \dot{W}_i = \dot{W}_{i-1} A_i + W_{i-1} \dfrac{\partial A_i}{\partial q_i} \dot{q}_i \\ \ddot{W}_i = \ddot{W}_{i-1} A_i + 2\dot{W}_{i-1} \dfrac{\partial A_i}{\partial q_i} \dot{q}_i + W_{i-1} \dfrac{\partial^2 A_i}{\partial q_i^2} \dot{q}_i^2 + W_{i-1} \dfrac{\partial A_i}{\partial q_i} \ddot{q}_i \\ \ddot{\overline{p}}_i = \ddot{\overline{p}}_{i-1} - \ddot{W}_i {}^i\overline{p}_i^* \qquad (i = 1, \ldots, n) \end{cases} \tag{2.17}$$

$$\begin{cases} e_i = e_{i+1} + m_i \ddot{\overline{p}}_i^T + {}^i\overline{n}_i^T \ddot{W}_i^T \\ D_i = A_{i+1} D_{i+1} + {}^i\overline{p}_{i+1} e_{i+1} + {}^i\overline{n}_i \ddot{\overline{p}}_i^T + \underline{\underline{J}}_i \ddot{W}_i^T \\ \overline{C}_i = m_i {}^i\overline{\gamma}_i + A_{i+1} \overline{C}_{i+1} \qquad (i = n, \ldots, 1) \end{cases} \tag{2.18}$$

where

${}^i\overline{p}_i^*$ --- The vector from the origin of the coordinate system i-1 to the origin of the coordinate system i expressed on the coordinate system i.

A_{i+1} --- The coordinate transformation matrix from the coordinate system i+1 to the coordinate system i.

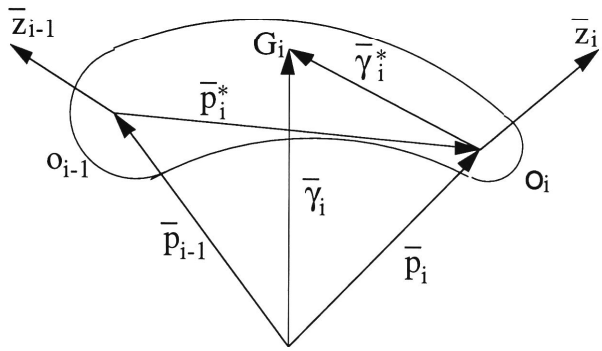

Fig. 2.3 The coordinates transformation

The Lagrange function is an energy function. It is clearly in the train of thought to build kinematics equation by the energy analysis, and the unknown constrained forces don't appear on the equations, and the recursive formulas are obtained. But comparing with the recursive Newton-Euler method, the efficiency of the method is still lower. By analysing causes, the following reasons are obtained. First, the first and second partial derivatives are still present in the recursive formulas and the recursive formulation is built on 3×3 matrices. So, there are still a large of unnecessary operations in the process of the computation. Second, the computation is carried out under the inertia coordinate system. So, the number of computation is large and the computational efficiency is lower. In addition, this method is only suitable for the inverse problem of dynamics and cannot be used for solving the direct problem of dynamics.

2.5. E. P. Popov (Е. П. Попов) Method

The algorithm suitable for both inverse and direct problems of dynamics was presented by Е. П. Попов on the basis of the Gaussian principle using the homogenous coordinates and optimisation method to describe and study the dynamics of manipulators.

The constrained quantity function of manipulators is as follows:

$$\zeta = \sum_{i=1}^{n} \{ tr[\frac{1}{2}\ddot{T}_i H_i \ddot{T}_i^T - \phi_i \ddot{T}_i^T] + \frac{1}{2} d_i \ddot{q}_i^2 - Q_i \ddot{q}_i \} \tag{2.19}$$

where

T_i --- The 4×4 coordinate transformation matrix from the coordinate system i to the inertia coordinate system.

H_i --- The 4×4 inertia matrix.

ϕ_i --- The 4×4 external force matrix acting on the link i.

d_i --- The inversion moment of inertia of the rotor of the ith driver.

Q_i --- The inversion value of the torque (force) of the ith driver.

The acceleration restrictive conditions of manipulators:

$$\ddot{T}_i = \ddot{T}_i A_i + B_i \ddot{q}_i + C_i \qquad (i = 1, \ldots, n) \tag{2.20}$$

The real accelerations of the system should not only satisfy the constrained conditions of the acceleration but also minimise the constrained quantity function according to the Gaussian principle. By applying the Lagrange multiplier method, the constrained minimisation problem of the function is converted into the unconstrained minimisation problem of the new function.

The new function can be expressed as following:

$$\zeta = \sum_{i=1}^{n}\{tr[\frac{1}{2}\ddot{T}_i H_i \ddot{T}_i^T - \phi_i \ddot{T}_i^T] + \sum_{j=1}^{n}\{[\frac{1}{2}d_j \ddot{q}_j^2 - Q_j \ddot{q}_j] + tr[\lambda_j(\ddot{T}_j - \ddot{T}_{j-1}A_j - B_j\ddot{q}_j - C_j)^T]\} \tag{2.21}$$

According to the necessary condition of the least minimum $\delta\zeta = 0$, one can obtain:

$$\sum_{i=0}^{n} tr\{(\ddot{T}_i H_i - \phi_i - \lambda_{i+1}A_{i+1}^T + \lambda_i)\delta\ddot{T}_i^T - \lambda_0 \delta\ddot{T}_0^T + \lambda_{n+1}A_{n+1}\delta\ddot{T}_i^T\}$$

$$+\sum_{j=1}^{n}[d_j\ddot{q}_j - Q_j - tr(\lambda_j B_j^T)]\delta\ddot{q}_j = 0$$

Hence, the following conditions can be obtained:

1) The dual equation conditions:

$$\lambda_i = \lambda_{i+1}A_{i+1}^T + \phi_i - \ddot{T}_i H_i \qquad (i = 0,\ 1,\ ..,\ n) \tag{2.22}$$

2) The force equal conditions:

$$d_j\ddot{q}_j - Q_j - tr(\lambda_j B_j^T) = 0 \qquad (j = 1,\ ...,\ n) \tag{2.23}$$

3) The boundary conditions:

$$tr\{\lambda_0 \delta\ddot{T}_0^T\} = 0, \qquad tr\{\lambda_{n+1}\delta\ddot{T}_{n+1}^T\} = 0 \tag{2.24}$$

The physical meaning of the dualty equations is that the equation i includes all forces acting on the link i and the inertia forces of the link i.

Once the boundary conditions are given from the equation (2.24), the dynamical problems of manipulators can be solved from the equations (2.22), (2.23) and the following recursive formulas:

$$\begin{cases} \dot{T}_i = \dot{T}_{i-1}A_i + B_i\dot{q}_i \\ \ddot{T}_i = \ddot{T}_{i-1}A_i + B_i\ddot{q}_i + C_i \end{cases} \tag{2.25}$$

$$\begin{cases} B_i = T_{i-1}\theta A_i \\ C_i = 2\dot{T}_{i-1}\theta A_i\dot{q}_i + T_{i-1}\theta^2 A_i\dot{q}_i^2 \end{cases} \tag{2.26}$$

where

θ --- The 4×4 projection matrix.

A_i --- The 4×4 coordinate transformation matrix.

1. The relative between the external force matrix ϕ_i and the resultant force and moment on link i.

$$\phi_i = \begin{bmatrix} \sum_v F_{1i}^v \rho_{1i}^v & \sum_v F_{1i}^v \rho_{2i}^v & \sum_v F_{1i}^v \rho_{3i}^v & \cdot & \sum_v F_{1i}^v \\ \sum_v F_{2i}^v \rho_{1i}^v & \sum_v F_{2i}^v \rho_{2i}^v & \sum_v F_{2i}^v \rho_{3i}^v & \cdot & \sum_v F_{2i}^v \\ \sum_v F_{3i}^v \rho_{1i}^v & \sum_v F_{3i}^v \rho_{2i}^v & \sum_v F_{3i}^v \rho_{3i}^v & \cdot & \sum_v F_{3i}^v \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & 0 \end{bmatrix}$$

In the above expression the force F_i is given with respect to the inertia coordinate system, and the radius vector ρ_i is given with respect to the relative coordinate system.

$$\text{Let} \quad \phi_i^0 = \sum_v F_i^v (\gamma_i^v)^T \tag{2.27}$$

In the equation, the γ_i and F_i are given with respect to the inertia coordinate system.

Then, the following relative can be obtained:

$$\phi_i^0 = \phi_i T_i^T \tag{2.28}$$

The resultant force and moment acting on the link i can be obtained from the external force matrix ϕ_i

Let $\tilde{F}_i^0 = \begin{pmatrix} \tilde{F}_{1i}^0 & \tilde{F}_{2i}^0 & \tilde{F}_{3i}^0 & \tilde{F}_{4i}^0 & \tilde{F}_{5i}^0 & \tilde{F}_{6i}^0 \end{pmatrix}^T$

and $\tilde{F}_i = \begin{pmatrix} \tilde{F}_{1i} & \tilde{F}_{2i} & \tilde{F}_{3i} & \tilde{F}_{4i} & \tilde{F}_{5i} & \tilde{F}_{6i} \end{pmatrix}^T$

are the projection of the resultant force and moment on the inertia coordinate system and the body coordinate system respectively. Therefore, all the information about the forces which are necessary for the rigid-body motion can be computed from the following equations.

$$\begin{cases} \tilde{F}_{ki}^0 = tr[\phi_i T_i^T \theta_k^T] \\ \tilde{F}_{ki} = tr[\phi_i \theta_k^T T_i^T] \qquad (k = 1, \ldots, 6) \end{cases} \tag{2.29}$$

When k=1, 2, 3, the moment can be computed, and when k=4, 5, 6, the force can be computed from the equation (2.29). The parameter θ_k is the projection matrix.

2. Inertia matrix H

The inertia matrix corresponding to the link i can be written as:

$$H_i = \begin{bmatrix} I_i & . & m_i \gamma_i \\ . & . & . \\ m_i \gamma_i^T & . & m_i \end{bmatrix}$$

where, m_i is the mass of link i, and γ_i is the coordinate vector of the mass centre of the link i:

$$\gamma_i = \begin{pmatrix} \gamma_{1i} & \gamma_{2i} & \gamma_{3i} \end{pmatrix}^T$$

The I_i is defined as:

$$I_i = \begin{bmatrix} \sum_v m_i^v (\gamma_{1i}^v)^2 & \sum_v m_i^v \gamma_{1i}^v \gamma_{2i}^v & \sum_v m_i^v \gamma_{1i}^v \gamma_{3i}^v \\ \sum_v m_i^v \gamma_{2i}^v \gamma_{1i}^v & \sum_v m_i^v (\gamma_{2i}^v)^2 & \sum_v m_i^v \gamma_{2i}^v \gamma_{3i}^v \\ \sum_v m_i^v \gamma_{3i}^v \gamma_{1i}^v & \sum_v m_i^v \gamma_{3i}^v \gamma_{2i}^v & \sum_v m_i^v (\gamma_{3i}^v)^2 \end{bmatrix}$$

The following are the relatives between the elements of I_i and the elements of the inertia tensor matrix J_i:

$$\begin{cases} I_i^{11} = (J_i^{22} + J_i^{33} - J_i^{11})/2 \\ I_i^{22} = (J_i^{11} + J_i^{33} - J_i^{22})/2 \\ I_i^{33} = (J_i^{11} + J_i^{22} - J_i^{33})/2 \\ I_i^{12} = I_i^{21} = J_i^{12} \\ I_i^{23} = I_i^{32} = J_i^{23} \\ I_i^{31} = I_i^{13} = J_i^{13} \end{cases} \tag{2.30}$$

To use the minimisation method to analyse the dynamically system is the advantage of the method, but the disadvantages are that the equation cannot be built and the much information of the system cannot be obtained. In addition, because the 4×4 matrices are introduced in the method, such as A_i, ϕ_i, H_i, the number of the matrices increases as the number of the links of manipulators increases, which makes the computation complicated and the number of computation raised.

CHAPTER 3

A NEW GENERAL DYNAMICAL MODEL FOR MANIPULATORS

3.1. Introduction

Along with the vigorous development of science and technology, modern industry has set some still higher requirements on robots. These requirements make robots develop toward high speed, heavy load applications, and more accurate. It makes the design and control of robots more complex. Modern industrial robots are not only the complicated executive mechanisms described by a system of equations of kinematics and dynamics, but also have to be operated coordinately with the power driving parts and control systems in order to ensure that their movements have a definite purpose. By using computers to simulate the dynamical characteristics of this kind of system as an efficient way for revealing reasonable motion schemes of mechanisms and improving the dynamical characteristics of mechanisms, the problems that arise on the phases of design, manufacture, test and operation can be explained properly.

On the control aspect, the dynamical real-time control for robots has become inevitable requirement. In order to ensure that robots move according to schedule locus, it is necessary to provide the inertia characteristic of systems and the influence of all applied forces over control coordinates for drive devices. To realise this kind of dynamical control, we have to compute the dynamics of robots. One of the important tasks of robot dynamics is to develop a method by which the dynamical model of highly nonlinear, coupled and multi-degrees of freedom for robot mechanisms can be

formed automatically and solved rapidly by computers in order to satisfy the requirement of control and simulation.

The manipulator consists of a group of rigid-bodies. It is a typical multi-rigid-body system. Manipulator is a kind of space mechanism system of open chain with multi-degrees of freedom. Most manipulators are simple chain structures with joints between links and are generally the lower-pair mechanisms. Owing to these characteristics, it is a kind of concrete and relatively simple multi-rigid-body system. Therefore, if we study it directly by the method of multi-rigid-body dynamics, we will make the simple problem more complicated. So, we should derive succinct and effective algorithms according to these characteristics of manipulators on the basis of the multi-rigid-body system dynamics when we study the modelling method of manipulator dynamics.

In recent years, many succinct and simple methods of robot dynamics have been derived according to the characteristics of manipulators worldwide. These can be roughly classified into three domains: numerical method, symbolical method and mixed method. In these methods the numerical methods that form and solve manipulator dynamical equations automatically have obtained great progress in past decade. According to the form of dynamical equations the numerical method can be classified into two types: recursive and nonrecursive. The recursive algorithms can avoid a large number of unnecessary repeated steps in dynamical calculation, which make the efficiency of dynamical calculation rise largely compared with the nonrecursive algorithms. To use the recursive method is an effective way of picking up computational speed and cutting down memory space. Useful research work on this aspect is found in References [25-31, 38-39].

In this chapter, a general form of dynamical equations for the manipulator with revolute and/or prismatic joints is presented according to structural characteristics of

the serial manipulators on the basis of multi-rigid-body system dynamics. The dynamical equations are vectorial and complete recursive and are suitable for general system of manipulators with revolute and/or prismatic joints. It suits the direct or inverse problems of dynamics .

3.2. Manipulator Mechanism

An open chain robot mechanism consists of a chain of n+l rigid links. The links are arranged such that link i is connected to a preceding link i-1 and a following link i+1. In manipulators, two types of joints exist, prismatic and revolute joints. The prismatic joints are such that the adjacent links translate linearly with respect to each other along the joint axis, while the revolute joints allow adjacent links to rotate with respect to each other about the joint axis. They are driven by some type of actuator. Therefore, the link i motion with reference to the link i-1 depends only on one variable, rotation θ_i or translation d_i. Generally, the robot base is considered to be link 0. The last link n carries a gripper (hand) or a tool (drill, pincer) and is called the end effector of the robot. The location of an object in space is determined by six degrees of freedom, three of which represent position and the other three orientation. If a task is performed in space without constraints, 6 degrees of freedom are necessary. But if the task is performed in a plane, only 3 degrees of freedom are necessary. Usually, a typical manipulator consists of 6 degrees of freedom. Hartenberg-Denavit 4×4 transformation matrices have been applied to the proposed system in this dissertation as a general description of the relationship existing between coordinates for manipulator systems.

3.3 Hartenberg-Denavit Matrix

Here, the Hartenberg-Denavit 4×4 transformation matrix is briefly reviewed and then the transformation matrices and their derivatives will follow. For applications, the positions and velocities of each point in the system is described by transformation matrices. As far as the characteristics of the transformation matrices are concerned, the recursive relations due to the nature of the sequence links exist in the manipulator system and improve the computational efficiency.

In this dissertation, the coordinate systems on the links of a robot are set up according to the Denavit and Hartenberg representation [40], and the links are numbered consecutively from 0 to n starting from the base of the robot to the hand tip.

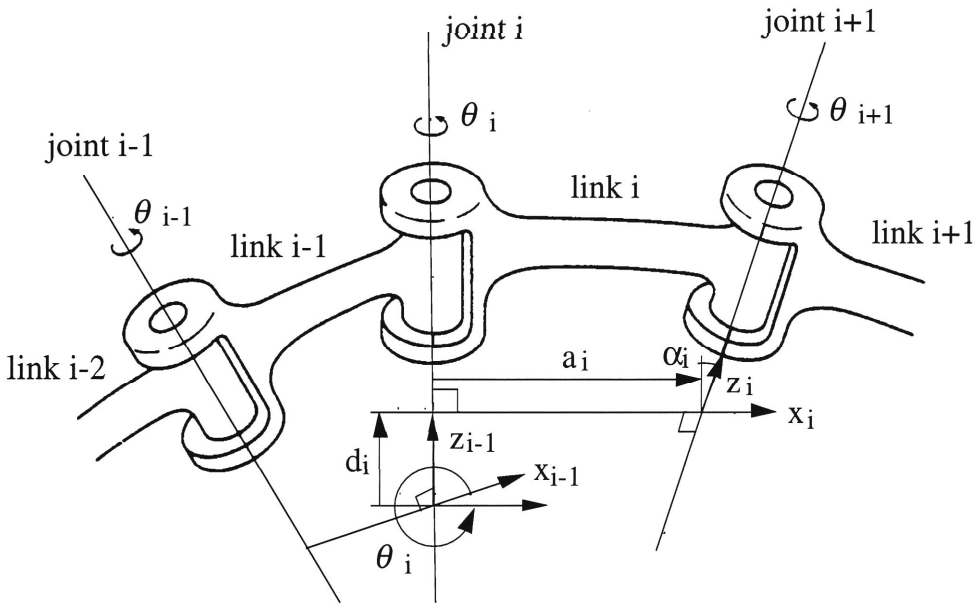

Fig. 3.1 Parameters relating adjacent link coordinate systems

Four parameters can be associated with every link of a manipulator as shown in Fig. 3.1. Each of these four parameters is defined with respect to the two joint axes attached to a particular link. The first two, a_i and α_i, define the structure of the link, while the second two, d_i and θ_i, determine the position of the neighbouring link. a_i is the common normal distance between the joint axes at each end of link i and is directed

from the axis of joint i to joint i+l. α_i, the angle between the two joint axes, is measured in a plane perpendicular to a_i. The angle is measured from axis i to axis i+l using a right-hand rule about a_i. The parameters a_i and α_i, called the length and twist of the link respectively, define the structure of the link. For a serial link configuration with two links and three joints, every joint axis has two normals to it, one for each of the links it connects. Thus the axis of joint i has both a_i and a_{i-1} perpendicular to it. The parameters d_i and θ_i are called the distance and angle between the links respectively, and are used to define the kinematic relationships. The relative position of the two links is given by d_i, which is the distance between the links i-l and i measured along the ith joint axis. θ_i is the angle between the links i-l and i measured in a plane perpendicular to the axis.

Of the four parameters in the D-H matrix, a_i and α_i are constants while either d_i or θ_i is the joint variable. Revolute joints have variables in terms of θ_i, while terms that are functions of a_i, α_i, and d_i are constants. In the case of a prismatic joint, a_i is zero, α_i and θ_i are constants, and d_i is the joint variable. Thus, for the linear joint all the entries of the D-H matrix, excluding d_i, are zero or constants.

According to the previous description, Let A_i be the Hartenberg-Denavit 4×4 transformation matrix from the coordinate system of link i to the coordinate system of link (i-1) in terms of four link parameters a_i, α_i, d_i and θ_i as follows:

$$A_i = \begin{bmatrix} R_{i-1,i} & . & \overline{p}_{i-1,i} \\ \cdots & . & \cdots \\ 0 & . & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & . & l_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & . & l_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & . & d_i \\ \cdots & \cdots & \cdots & . & \cdots \\ 0 & 0 & 0 & . & 1 \end{bmatrix} \qquad (3.1)$$

The transformation matrix relates points defined in frame i to frame i-1. By defining the coordinate frames and determining the entries for the D-H matrices, the position of the tool or gripper (last link and frame) with respect to the base (0 link and joint 1) as a function of the joint variables can be determined.

Where $R_{i-1,i}$ is a 3×3 rotation (transformation) matrix from the coordinate system of link i to the coordinate system of link i-1; $\overline{p}_{i-1,i}$ is a 3×1 position vector of the origin of the coordinate system of link i in the Coordinate system of link i-1.

3.4 Coordinate Transformation

Let T_i be the 4×4 D-H matrix from the coordinate system i to the inertia coordinate system, then

$$T_i = \begin{bmatrix} R_i & . & \overline{p}_i \\ \dots & . & \dots \\ 0 & . & 1 \end{bmatrix} = A_1 A_2 \dots A_i \tag{3.2}$$

where R_i is a 3×3 rotation (transformation) matrix from the coordinate system of link i to the base coordinate system and $\overline{p}_i$ is a 3×1 position vector of the origin of the coordinate system of link i in the base coordinate system. They satisfy the following forward recursions:

$$R_i = R_{i-1} R_{i-1,i}, \qquad R_1 = R_{0,1} \tag{3.3}$$

$$\overline{p}_i = \overline{p}_{i-1} + R_{i-1,i} \overline{p}_{i-1,i}, \qquad \overline{p}_1 = \overline{p}_{0,1} \tag{3.4}$$

Note that $\overline{p}_i$ is, at the moment, referenced to the base coordinate system. It can be transformed to its own link coordinate system. Thus,

$$\bar{p}_{i,i} = R_i^T \bar{p}_i$$
$$= \begin{cases} R_{i-1,i}^T \bar{p}_{i-1,i-1} + \left[d_i \quad a_i \sin\theta_i \quad a_i \cos\theta_i \right]^T, & \textit{for revolute } i \\ R_{i-1,i}^T (\bar{p}_{i-1,i-1} + \bar{p}_{i-1,i}), & \textit{for prismatic } i \end{cases} \quad (3.5)$$

Throughout this dissertation, 1) the symbol q_i denotes the joint variable for joint i whether it is θ_i or d_i; 2) $\hat{u}$ is a 3×3 skew symmetric tensor corresponding to the 3×1 vector $\bar{u} = \left[u_x \quad u_y \quad u_z \right]^T$ where [.]T denotes the transpose of [.] and is defined as

$$\hat{u} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \quad (3.6)$$

where "^" can be considered as an operator; and 3) $\tilde{W}$ is a 3×1 vector corresponding to the 3×3 matrix $W = \left[W_{ij} \right]$ and is defined as

$$\tilde{W} = \left[W_{23} - W_{32} \quad W_{31} - W_{13} \quad W_{12} - W_{21} \right]^T \quad (3.7)$$

where W_{ij} (i, j=1, 2, 3) denotes the (i, j)-th element of matrix W and "~" can be considered as an operator. Using equations (3.6) and (3.7), we have the following two important identities for the operators "^" and "~":

$$\hat{u}\bar{v} = \bar{u} \times \bar{v} \quad (3.8)$$

and

$$Trace(\hat{u}W) = \bar{u}^T \tilde{W} \quad (3.9)$$

where "×" denotes the vector cross product; $\bar{u}$ and $\bar{v}$ are two 3×1 vectors and W is a 3×3 matrix.

3.5 Kinematics of Manipulators

The kinematics of manipulators is described in this section, the first step in the dynamic modelling of any mechanical system is to establish the kinematical relationships and be able to define fundamental vector quantities viz., position, velocity, and acceleration. Modal analysis is used to incorporate the deflection of the link.

The kinematics of lower-pair mechanisms with rigid links has been completely described using the symbolic notation of the Hartenberg-Denavit Matrix. This 4×4 transformation matrix technique has been applied to develop the dynamic model for spatial linkages. The kinematics described by the Hartenberg-Denavit Matrix are straightforward and systematic for manipulators with rigid links.

Fig. 3.1 shows a general serial manipulator system consisting of n+1 links (rigid-bodies). The base coordinate system ($\overline{X}_1, \overline{Y}_1, \overline{Z}_1$) attached to link 0 with origin O_1. Links and joints are numbered from 1 to n starting from the base (link 0). The body coordinate system i+1 ($\overline{X}_{i+1}, \overline{Y}_{i+1}, \overline{Z}_{i+1}$) is attached to link i with origin O_{i+1}. The ith joint is situated between links i and i-1. The type indicative symbol is S_i.: S_i.=0 when the joint i is hinge; S_i=1 when the joint i is prismatic joint. The unit vector coincides with the rotating axis of joint i when S_i=0 and with the sliding axis of joint i when S_i=1. The generalised coordinate q_i is the relative rotation angle θ_i of joint i when S_i=0 and the relative displacement d_i of joint i when S_i=1. The point O_i coincides with the point O_i' on the rotating axis i. The point O_i is at the link i-1 and the point O_i' is at the link i when S_i=1. The point G_i is the mass centre of link i. The mass of link i is m_i and its central inertia tensor is $\underline{\underline{J}}_i$. The $\overline{\gamma}_i$ is the position vector of the mass centre of link i relative to the origin O of base coordinate system. The angular velocity $\overline{\omega}_i$ can be written as $\dot{\overline{\pi}}_i$. $\overline{\pi}_i$ is defined as the virtual rotative angle.

We define these vectors as follows (Fig. 1):

$$\bar{d}_{ii} = \overline{O_i G_i}; \qquad \bar{d}_{i+1,i} = \overline{O_{i+1,} G_i}$$

$$\bar{p}_i = \overline{O_i' O'}_{i+1} = (\bar{d}_{ii} - \bar{d}_{i+1,i}) + S_{i+1} q_{i+1} \bar{z}_{i+1}$$

$$\bar{d}_{ij} = \overline{O_i G_j} = \bar{p}_i + \bar{d}_{i+1,j.}$$

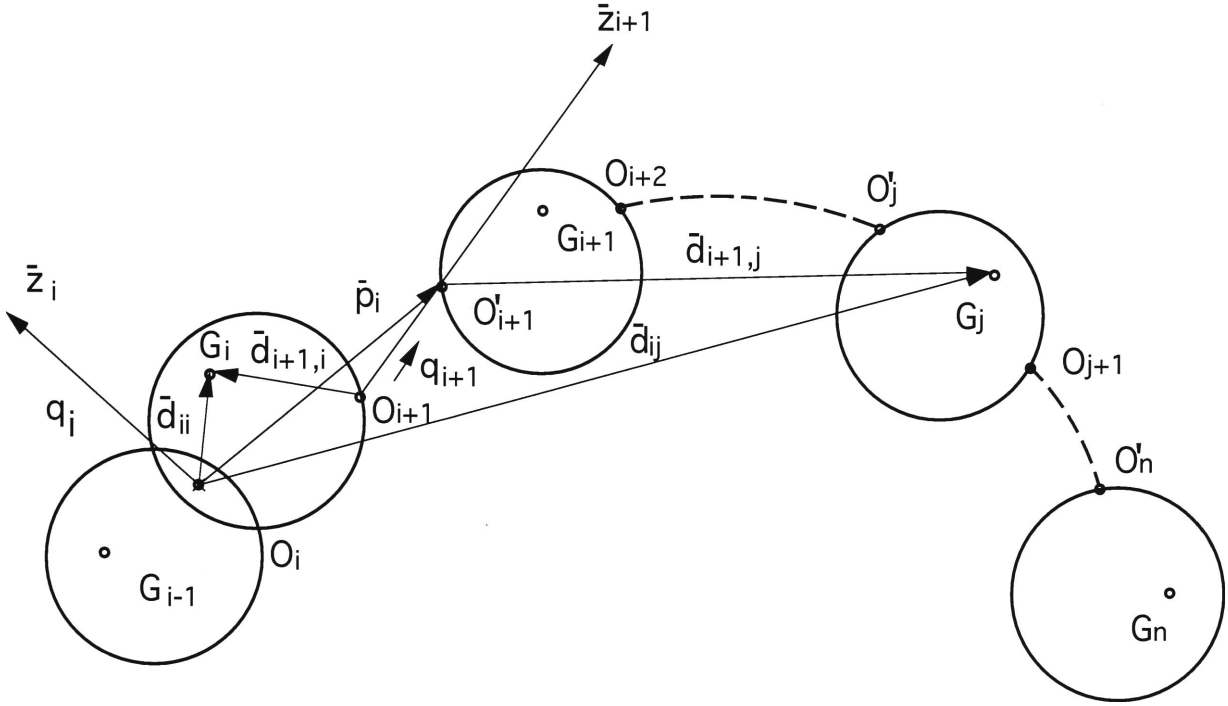

Fig. 3.2 The schematic diagram of a general n-link serial robot manipulator and the relationship of defined quantities to the links.

According to the characteristics of the serial manipulator, the angular velocity of link i and the linear velocity at the mass centre of link i can be given as follows by making use of the superposition theorem. (let $S_i' = 1 - S_i$)

$$\bar{\omega}_i = \sum_{k=1}^{i} S_k' \bar{z}_k \dot{q}_k \tag{3.10}$$

$$\bar{v}_i = \sum_{k=1}^{i} S_k' \dot{q}_k (\bar{z}_k \times \bar{d}_{ki}) + \sum_{k=1}^{i} S_k \dot{q}_k \bar{z}_k \tag{3.11}$$

Thus, the variation can be obtained from Eqs. (3.10) and (3.11):

$$\delta\bar{\pi}_i = \sum_{k=1}^{i} S'_k \bar{z}_k \delta q_k \tag{3.12}$$

$$\delta\bar{\gamma}_i = \sum_{k=1}^{i} S'_k (\bar{z}_k \times \bar{d}_{ki}) \delta q_k + \sum_{k=1}^{i} S_k \bar{z}_k \delta q_k \tag{3.13}$$

The angular acceleration of link i and the linear acceleration at the mass centre of link i can be derived from Eqs. (3.10) and (3.11) by differentiating with respect to time.

$$\dot{\bar{\omega}}_i = \sum_{k=1}^{i} S'_k \ddot{q}_k \bar{z}_k + \sum_{k=1}^{i} \sum_{j=1}^{k-1} (\bar{z}_j \times \bar{z}_k) S'_k S'_j \dot{q}_k \dot{q}_j \tag{3.14}$$

$$\dot{\bar{v}}_i = \sum_{k=1}^{i} [S'_k (\bar{z}_k \times \bar{d}_{ki}) + S_k \bar{z}_k] \ddot{q} + \sum_{k=1}^{i} \sum_{j=1}^{k-1} \{[(\bar{z}_j \times \bar{z}_k) \times \bar{d}_{ki}] S'_k S'_j$$

$$+ (\bar{z}_j \times \bar{z}_k) S_k S'_j \} \dot{q}_j \dot{q}_k + \sum_{k=1}^{i} \sum_{j=k}^{i} \{[\bar{z}_k \times (\bar{z}_j \times \bar{d}_{ji})] S'_j S'_k + (\bar{z}_k \times \bar{z}_j) S_j S'_k \} \dot{q}_j \dot{q}_k \tag{3.15}$$

3.6 General Form of Dynamical Equations for the Manipulator with Revolute and/or Prismatic Joints

According to the D'Alembert's principle and the virtual work theorem, the general equation can be written as:

$$\sum_{i+1}^{n} \{(m_i \ddot{\bar{\gamma}}_i - \bar{F}_i) \cdot \delta\bar{\gamma}_i + [\underline{\underline{J}}_i \dot{\bar{\omega}}_i + \bar{\omega}_i \times (\underline{\underline{J}}_i \cdot \bar{\omega}_i) - \bar{M}_i] \cdot \delta\bar{\pi}_i \} = 0 \tag{3.16}$$

here,

$\ddot{\bar{\gamma}}_i$--- the acceleration at the mass centre of link i;

$\bar{F}_i, \bar{M}_i$--- the total external force and moment with respect to mass centre exerted on link i;

Combining Eqs.(3.10)--(3.15) and substituting them into Eq.(3.16), we can obtain:

$$\sum_{m=1}^{n}\{\sum_{k=1}^{n}\sum_{i=\max(m,k)}^{n}\{[\bar{z}_m\cdot\underline{\underline{J}}_i\cdot\bar{z}_k+m_i(\bar{z}_k\times\bar{d}_{ki})\cdot(\bar{z}_m\times\bar{d}_{mi})]S'_kS'_m+m_i(\bar{z}_k\times\bar{d}_{ki})\cdot\bar{z}_mS'_kS_m$$

$$+m_i\bar{z}_k\cdot\bar{z}_mS_kS_m+m_i(\bar{z}_m\times\bar{d}_{mi})\cdot\bar{z}_kS_kS'_m\}\ddot{q}_k+\sum_{k=1}^{n}\sum_{j=1}^{k-1}\sum_{i=max(m,k)}^{n}\{\{[(\bar{z}_j\times\bar{z}_k)\times\bar{d}_{ki}]\cdot(\bar{z}_m\times\bar{d}_{mi})m_i$$

$$+\bar{z}_m\cdot\underline{\underline{J}}_i\cdot(\bar{z}_j\times\bar{z}_k)\}S'_mS'_kS'_j+[(\bar{z}_j\times\bar{z}_k)\times\bar{d}_{ki}]\cdot\bar{z}_mS'_kS'_jS_mm_i+(\bar{z}_j\times\bar{z}_k)\cdot(\bar{z}_m\times\bar{d}_{mi})S'_jS_kS'_mm_i$$

$$+(\bar{z}_j\times\bar{z}_k)\cdot\bar{z}_mS_kS'_jS_mm_i\}\dot{q}_j\dot{q}_k+\sum_{k=1}^{n}\sum_{j=k}^{n}\sum_{i=max(m,k,j)}^{n}\{[\bar{z}_k\times(\bar{z}_j\times\bar{d}_{ji})]\cdot(\bar{z}_m\times\bar{d}_{mi})S'_jS'_kS'_mm_i$$

$$+[\bar{z}_k\times(\bar{z}_j\times\bar{d}_{ji})]\cdot\bar{z}_mS'_jS'_kS_mm_i+(\bar{z}_k\times\bar{z}_j)\cdot(\bar{z}_m\times\bar{d}_{mi})S_jS'_kS'_mm_i$$

$$+(\bar{z}_k\times\bar{z}_j)\cdot\bar{z}_mS'_kS_jS_mm_i\}\dot{q}_k\dot{q}_j+\sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{i=max(m,k,j)}^{n}\bar{z}_m\cdot\bar{z}_j\times(\underline{\underline{J}}_i\cdot\bar{z}_k)S'_kS'_jS'_m\dot{q}_k\dot{q}_j$$

$$-\sum_{i=m}^{n}(\bar{z}_m\times\bar{d}_{mi})\cdot\bar{F}_iS'_m-\sum_{i=m}^{n}\bar{z}_m\cdot\bar{F}_iS_m-\sum_{i=m}^{n}\bar{M}_i\cdot\bar{z}_mS'_m\}\delta q_m=0$$

Because the δq_m are independent variations, all of them cannot be zero at the same time. So we have:

$$\sum_{k=1}^{n}\sum_{i=max(m,k)}^{n}\{[\bar{z}_m\cdot\underline{\underline{J}}_i\bar{z}_k+m_i(\bar{z}_k\times\bar{d}_{ki})\cdot(\bar{z}_m\times\bar{d}_{mi})]S'_kS'_m+m_i(\bar{z}_k\times\bar{d}_{ki})\cdot\bar{z}_mS'_kS_m$$

$$+m_i\bar{z}_k\cdot\bar{z}_mS_kS_m+m_i(\bar{z}_m\times\bar{d}_{mi})\cdot\bar{z}_kS_kS'_m\}\ddot{q}_k+\sum_{k=1}^{n}\sum_{j=1}^{k-1}\sum_{i=max(m,k)}^{n}\{\{[(\bar{z}_j\times\bar{z}_k)\times\bar{d}_{ki}]\cdot(\bar{z}_m\times\bar{d}_{mi})m_i$$

$$+\bar{z}_m\cdot\underline{\underline{J}}_i\cdot(\bar{z}_j\times\bar{z}_k)\}S'_mS'_kS'_j+[(\bar{z}_j\times\bar{z}_k)\times\bar{d}_{ki}]\cdot\bar{z}_mS'_kS'_jS_mm_i+(\bar{z}_j\times\bar{z}_k)\cdot(\bar{z}_m\times\bar{d}_{mi})S'_jS_kS'_mm_i$$

$$+(\bar{z}_j\times\bar{z}_k)\cdot\bar{z}_mS_kS'_jS_mm_i\}\dot{q}_j\dot{q}_k+\sum_{k=1}^{n}\sum_{j=k}^{n}\sum_{i=max(m,k,j)}^{n}\{[\bar{z}_k\times(\bar{z}_j\times\bar{d}_{ji})]\cdot(\bar{z}_m\times\bar{d}_{mi})S'_jS'_kS'_mm_i$$

$$+[\bar{z}_k\times(\bar{z}_j\times\bar{d}_{ji})]\cdot\bar{z}_mS'_jS'_kS_mm_i+(\bar{z}_k\times\bar{z}_j)\cdot(\bar{z}_m\times\bar{d}_{mi})S_jS'_kS'_mm_i$$

$$+(\bar{z}_k\times\bar{z}_j)\cdot\bar{z}_mS'_kS_jS_mm_i\}\dot{q}_k\dot{q}_j+\sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{i=max(m,k,j)}^{n}\bar{z}_m\cdot\bar{z}_j\times(\underline{\underline{J}}_i\cdot\bar{z}_k)S'_kS'_jS'_m\dot{q}_k\dot{q}_j$$

$$-\sum_{i=m}^{n}(\bar{z}_m\times\bar{d}_{mi})\cdot\bar{F}_iS'_m-\sum_{i=m}^{n}\bar{z}_m\cdot\bar{F}_iS_m-\sum_{i=m}^{n}\bar{M}_i\cdot\bar{z}_mS'_m=0$$

(m=1, 2, ..., ..., n) (3.16*)

In view of $(\bar{a}\times\bar{b})\cdot(\bar{c}\times\bar{d})=-\bar{a}\cdot(\hat{\underline{\underline{b}}}\cdot\hat{\underline{\underline{d}}})\cdot\bar{c}$, where $\hat{\underline{\underline{b}}}$ and $\hat{\underline{\underline{d}}}$ are the skew symmetrical tensor of vector $\bar{b}$ and $\bar{d}$ respectively. So the first item of Eq.(16*) can be written as:

$$\sum_{k=1}^{n}\sum_{i=max(m,k)}^{n}\{[\bar{z}_m\cdot\underline{\underline{J}}_i\bar{z}_k+m_i(\bar{z}_k\times\bar{d}_{ki})\cdot(\bar{z}_m\times\bar{d}_{mi})]S'_kS'_m+m_i(\bar{z}_k\times\bar{d}_{ki})\cdot\bar{z}_mS'_kS_m$$

$$+m_i\bar{z}_k\cdot\bar{z}_mS_kS_m+m_i(\bar{z}_m\times\bar{d}_{mi})\cdot\bar{z}_kS_kS'_m\}\ddot{q}_k$$

$$=\sum_{k=1}^{n}\sum_{i=max(m,k)}^{n}\{\bar{z}_m\cdot[\underline{\underline{J}}_i-m_i(\hat{\underline{\underline{d}}}_{mi}\cdot\hat{\underline{\underline{d}}}_{ki})]\cdot\bar{z}_kS'_kS'_m+m_i(\bar{z}_k\times\bar{d}_{ki})\cdot\bar{z}_mS'_kS_m$$

$$+m_i\bar{z}_k\cdot\bar{z}_mS_kS_m+m_i(\bar{z}_m\times\bar{d}_{mi})\cdot\bar{z}_kS_kS'_m\}\ddot{q}_k$$

$$=\sum_{k=1}^{n}\{\bar{z}_m\cdot\underline{\underline{J}}^{mk}\cdot\bar{z}_kS'_kS'_m+\sum_{i=max(m,k)}^{n}[m_i(\bar{z}_k\times\bar{d}_{ki})\cdot\bar{z}_mS'_kS_m+m_i\bar{z}_k\cdot\bar{z}_mS_kS_m$$

$$+m_i(\bar{z}_m\times\bar{d}_{mi})\cdot\bar{z}_kS_kS'_m]\}\ddot{q}_k$$

$$=\sum_{k=1}^{n}a_{mk}\ddot{q}_k$$

where,

$$a_{mk}=\bar{z}_m\cdot\underline{\underline{J}}^{mk}\cdot\bar{z}_kS'_kS'_m+\sum_{i=max(m,k)}^{n}[m_i(\bar{z}_k\times\bar{d}_{ki})\cdot\bar{z}_mS'_kS_m+m_i\bar{z}_k\cdot\bar{z}_mS_kS_m$$

$$+m_i(\bar{z}_m\times\bar{d}_{mi})\cdot\bar{z}_kS_kS'_m] \qquad (a)$$

$$\underline{\underline{J}}^{mk}=\sum_{i=max(m,k)}^{n}[\underline{\underline{J}}_i-m_i(\hat{\underline{\underline{d}}}_{mi}\cdot\hat{\underline{\underline{d}}}_{ki})] \qquad (b)$$

Similarly, the second item of Eq.(3.16*) can be written as:

$$\sum_{k=1}^{n}\sum_{j=1}^{k-1}\sum_{i=max(m,k)}^{n}\{\{[(\bar{z}_j\times\bar{z}_k)\times\bar{d}_{ki}]\cdot(\bar{z}_m\times\bar{d}_{mi})m_i+\bar{z}_m\cdot\underline{\underline{J}}_i\cdot(\bar{z}_j\times\bar{z}_k)\}S'_mS'_kS'_j+[(\bar{z}_j\times\bar{z}_k)$$

$$\times\bar{d}_{ki}]\cdot\bar{z}_mS'_kS'_jS_m m_i+(\bar{z}_j\times\bar{z}_k)\cdot(\bar{z}_m\times\bar{d}_{mi})S'_jS_kS'_m m_i+(\bar{z}_j\times\bar{z}_k)\cdot\bar{z}_mS_kS'_jS_m m_i\}\dot{q}_j\dot{q}_k$$

$$=\sum_{k=1}^{n}\sum_{j=1}^{k-1}\{\bar{z}_m\cdot\underline{\underline{J}}^{mk}\cdot(\bar{z}_j\times\bar{z}_k)S'_mS'_kS'_j+\sum_{i=max(m,k)}^{n}\{[(\bar{z}_j\times\bar{z}_k)\times\bar{d}_{ki}]\cdot\bar{z}_mS_mS'_kS'_jm_i$$

$$+(\bar{z}_j\times\bar{z}_k)\cdot(\bar{z}_m\times\bar{d}_{mi})S'_jS_kS'_m m_i+(\bar{z}_j\times\bar{z}_k)\cdot\bar{z}_mS_kS'_jS_m m_i\}\dot{q}_j\dot{q}_k$$

$$=\sum_{k=1}^{n}\sum_{j=1}^{k-1}H_{mkj}\dot{q}_j\dot{q}_k$$

where,

$$H_{mkj} = \bar{z}_m \cdot \underline{\underline{J}}^{mk} \cdot (\bar{z}_j \times \bar{z}_k) S'_m S'_k S'_j + \sum_{i=max(m,k)}^{n} \{ [(\bar{z}_j \times \bar{z}_k) \times \bar{d}_{ki}] \cdot \bar{z}_m S_m S'_k S'_j m_i$$
$$+ (\bar{z}_j \times \bar{z}_k) \cdot (\bar{z}_m \times \bar{d}_{mi}) S'_j S_k S'_m m_i + (\bar{z}_j \times \bar{z}_k) \cdot \bar{z}_m S_k S'_j S_m m_i \} \qquad \text{(d)}$$

The third item of Eq.(3.16*) is:

$$\sum_{k=1}^{n} \sum_{j=k}^{n} \sum_{i=max(m,k,j)}^{n} \{ [\bar{z}_k \times (\bar{z}_j \times \bar{d}_{ji})] \cdot (\bar{z}_m \times \bar{d}_{mi}) S'_j S'_k S'_m m_i$$
$$+ [\bar{z}_k \times (\bar{z}_j \times \bar{d}_{ji})] \cdot \bar{z}_m S'_j S'_k S_m m_i + (\bar{z}_k \times \bar{z}_j) \cdot (\bar{z}_m \times \bar{d}_{mi}) S_j S'_k S'_m m_i$$
$$+ (\bar{z}_k \times \bar{z}_j) \cdot \bar{z}_m S'_k S_j S_m m_i \} \dot{q}_k \dot{q}_j$$
$$= \sum_{k=1}^{n} \sum_{j=k}^{n} \sum_{i=max(m,k,j)}^{n} [(\bar{z}_k \times \bar{c}_{ji}) \cdot \bar{c}_{mi} S'_j S'_k S'_m m_i + (\bar{z}_k \times \bar{c}_{ji}) \cdot \bar{z}_m S'_j S'_k S_m m_i$$
$$+ (\bar{z}_k \times \bar{z}_j) \cdot \bar{c}_{mi} S_j S'_k S'_m m_i + (\bar{z}_k \times \bar{z}_j) \cdot \bar{z}_m S_m S'_k S_j m_i] \dot{q}_k \dot{q}_j$$
$$= \sum_{k=1}^{n} \sum_{j=k}^{n} C_{mkj} \dot{q}_j \dot{q}_k$$

where,

$$C_{mkj} = \sum_{i=max(m,k,j)}^{n} [(\bar{z}_k \times \bar{c}_{ji}) \cdot \bar{c}_{mi} S'_j S'_k S'_m m_i + (\bar{z}_k \times \bar{c}_{ji}) \cdot \bar{z}_m S'_j S'_k S_m m_i$$
$$+ (\bar{z}_k \times \bar{z}_j) \cdot \bar{c}_{mi} S_j S'_k S'_m m_i + (\bar{z}_k \times \bar{z}_j) \cdot \bar{z}_m S_m S'_k S_j m_i] \qquad \text{(e)}$$

$$\bar{c}_{mi} = \bar{z}_m \times \bar{d}_{mi} \qquad \text{(f)}$$

The fourth item of Eq.(3.16*) is:

$$\sum_{k=1}^{n} \sum_{j=1}^{n} \sum_{i=max(m,k,j)}^{n} \bar{z}_m \cdot \bar{z}_j \times (\underline{\underline{J}}_i \cdot \bar{z}_k) S'_k S'_j S'_m \dot{q}_k \dot{q}_j = \sum_{k=1}^{n} \sum_{j=1}^{n} D_{mkj} \dot{q}_k \dot{q}_j$$

where,

$$D_{mkj} = \sum_{i=max(m,k,j)}^{n} \bar{z}_m \cdot \bar{z}_j \times (\underline{\underline{J}}_i \cdot \bar{z}_k) S'_k S'_j S'_m \qquad \text{(g)}$$

For every link we consider only the gravity and driving force. Let τ_i is the driving force (prismatic joint) or the driving moment (revolute joint). We have:

$$\overline{F}_i = \begin{cases} \tau_i S_i \bar{z}_i - \tau_{i+1} S_{i+1} \bar{z}_{i+1} + m_i \bar{g} & i < n \\ \tau_n S_n \bar{z}_n + m_n \bar{g} & i = n \end{cases}$$

$$M_i = \begin{cases} \tau_i S_i' \bar{z}_i - \tau_{i+1} S_{i+1}' \bar{z}_{i+1} + (\bar{d}_{ii} \times \bar{z}_i)\tau_i S_i - (d_{i+1,i} \times \bar{z}_{i+1})\tau_{i+1} S_{i+1} & i < n \\ \tau_n S_n' \bar{z}_n + (\bar{d}_{nn} \times \bar{z}_n)\tau_n S_n & i = n \end{cases}$$

where,

$$\bar{g} = -g\bar{z}_1$$

So the fifth item of Eq.(3.16*) can be written as:

$$\sum_{i=m}^{n} (\bar{z}_m \times \bar{d}_{mi}) \cdot \overline{F}_i S_m' = \sum_{i=m}^{n} \bar{c}_{mi} \cdot (\tau_i S_i \bar{z}_i - \tau_{i+1} S_{i+1} \bar{z}_{i+1}) S_m' + (\bar{z}_m \times \overline{U}^m) \cdot g\bar{z}_1 S_m'$$

where,

$$\overline{U}^m = \sum_{i=m}^{n} m_i \bar{d}_{mi} \tag{h}$$

The sixth item of Eq.(3.16*) is:

$$\begin{aligned} &\sum_{i=m}^{n} \bar{z}_m \cdot \overline{F}_i S_m = (\bar{z}_m \cdot \tau_m \bar{z}_m S_m S_m - \bar{z}_m \cdot \tau_{m+1} \bar{z}_{m+1} S_{m+1} S_m + \bar{z}_m \cdot m_m \bar{g} S_m) \\ &+(\bar{z}_m \cdot \tau_{m+1} \bar{z}_{m+1} S_{m+1} S_m - \bar{z}_m \cdot \tau_{m+2} \bar{z}_{m+2} S_{m+2} S_m + \bar{z}_m \cdot m_{m+1} \bar{g} S_m + \cdots \\ &+(\bar{z}_m \cdot \tau_{n-1} \bar{z}_{n-1} S_{n-1} S_m - \bar{z}_m \cdot \tau_n \bar{z}_n S_n S_m + \bar{z}_m \cdot m_{n-1} \bar{g} S_m) \\ &+\bar{z}_m \cdot \tau_n S_n \bar{z}_n S_m + \bar{z}_m \cdot m_n \bar{g} S_m \\ &= \tau_m S_m + \bar{z}_m \cdot M^m \bar{g} S_m \end{aligned}$$

where:

$$M^m = \sum_{i=m}^{n} m_i \tag{i}$$

The seventh item of Eq.(3.16*) is:

$$\sum_{i=m}^{n} \overline{M}_i \cdot \bar{z}_m S_m' = \tau_m S_m' + S_m' \sum_{i=m+1}^{n} (\bar{d}_{ii} - \bar{d}_{i,i-1}) \times \bar{z}_i \cdot \bar{z}_m \tau_i S_i$$

We note $\tau_m S_m' + \tau_m S_m = \tau_m$. Combine the fifth, sixth and seventh items of Eq.(3.16*), we have:

$$\sum_{i=m}^{n}[(\bar{z}_m \times \bar{d}_{mi}) \cdot \bar{F}_i S'_m + \bar{z}_m \cdot \bar{F}_i S_m + \bar{M}_i \cdot \bar{z}_m S'_m]$$
$$= \tau_m - E_m$$

where,

$$-E_m = S'_m[\bar{z}_m \cdot \sum_{i=m+1}^{n}(\bar{d}_{ii} - \bar{d}_{i,i-1}) \times \bar{z}_i \tau_i S_i + \sum_{i=m}^{n} \bar{c}_{mi} \cdot (\tau_i S_i \bar{z}_i - \tau_{i+1} S_{i+1} \bar{z}_{i+1})$$
$$+(\bar{z}_m \times \bar{U}^m) \cdot \bar{g}] + S_m M^m \bar{z}_m \cdot \bar{g} \qquad \text{(j)}$$

So that the Eq. (3.16*) can be written as:

$$\tau_m = \sum_{k=1}^{n} a_{mk} \ddot{q}_k + \sum_{k=1}^{n}\sum_{j=1}^{n} b_{mkj} \dot{q}_j \dot{q}_k + E_m \qquad \text{(m=1,2,... ...,n)} \qquad (3.17)$$

where,

$$a_{mk} = \bar{z}_m \cdot \underline{\underline{J}}^{mk} \cdot \bar{z}_k S'_k S'_m + \sum_{i=\max(m,k)}^{n}[m_i(\bar{z}_k \times \bar{d}_{ki}) \cdot \bar{z}_m S'_k S_m + m_i \bar{z}_k \cdot \bar{z}_m S_k S_m$$
$$+m_i(\bar{z}_m \times \bar{d}_{mi}) \cdot \bar{z}_k S_k S'_m] \qquad \text{(a)}$$

$$\underline{\underline{J}}^{mk} = \sum_{i=\max(m,k)}^{n}[\underline{\underline{J}}_i - m_i(\hat{\underline{\underline{d}}}_{mi} \cdot \hat{\underline{\underline{d}}}_{ki})] \qquad \text{(b)}$$

$$b_{mkj} = \begin{cases} H_{mkj} + D_{mkj} & j < k \\ C_{mkj} + D_{mkj} & j \geq k \end{cases} \qquad \text{(c)}$$

$$H_{mkj} = \bar{z}_m \cdot \underline{\underline{J}}^{mk} \cdot (\bar{z}_j \times \bar{z}_k) S'_m S'_k S'_j + \sum_{i=\max(m,k)}^{n}\{[(\bar{z}_j \times \bar{z}_k) \times \bar{d}_{ki}] \cdot \bar{z}_m S_m S'_k S'_j m_i$$
$$+(\bar{z}_j \times \bar{z}_k) \cdot (\bar{z}_m \times \bar{d}_{mi}) S'_j S_k S'_m m_i + (\bar{z}_j \times \bar{z}_k) \cdot \bar{z}_m S_k S'_j S_m m_i\} \qquad \text{(d)}$$

$$C_{mkj} = \sum_{i=max(m,k,j)}^{n}[(\bar{z}_k \times \bar{c}_{ji}) \cdot \bar{c}_{mi} S'_j S'_k S'_m m_i + (\bar{z}_k \times \bar{c}_{ji}) \cdot \bar{z}_m S'_j S'_k S_m m_i$$
$$+(\bar{z}_k \times \bar{z}_j) \cdot \bar{c}_{mi} S_j S'_k S'_m m_i + (\bar{z}_k \times \bar{z}_j) \cdot \bar{z}_m S_m S'_k S_j m_i] \qquad \text{(e)}$$

$$\bar{c}_{mi} = \bar{z}_m \times \bar{d}_{mi} \qquad \text{(f)}$$

$$D_{mkj} = \sum_{i=\max(m,k,j)}^{n} \bar{z}_m \cdot \bar{z}_j \times (\underline{\underline{J}}_i \cdot \bar{z}_k) S'_k S'_j S'_m \qquad \text{(g)}$$

$$\bar{U}^m = \sum_{i=m}^{n} m_i \bar{d}_{mi} \qquad \text{(h)}$$

$$M^m = \sum_{i=m}^{n} m_i \qquad \text{(i)}$$

$$-E_m = S'_m[\bar{z}_m \cdot \sum_{i=m+1}^{n} (\bar{d}_{ii} - \bar{d}_{i,i-1}) \times \bar{z}_i \tau_i S_i + \sum_{i=m}^{n} \bar{c}_{mi} \cdot (\tau_i S_i \bar{z}_i - \tau_{i+1} S_{i+1} \bar{z}_{i+1})$$
$$+(\bar{z}_m \times \bar{U}^m) \cdot \bar{g}] + S_m M^m \bar{z}_m \cdot \bar{g} \qquad \text{(j)}$$

Previously undefined symbols are:

a_{mk}--- the elements of symmetric inertia matrix [A].

b_{mkj}---the elements of centrifugal and coriolis effects matrix [B].

E_m---the elements of gravity effects column matrix [E].

τ_m---the torque exerted by actuator at joint m.

$\hat{\underline{\underline{d}}}_{mi}$---the skew symmetrical tensor of vector $\bar{d}_{mi}$.

This Eq.(3.17) is the General Form of dynamical equations of serial manipulator that we have derived.

We will discuss the properties of coefficients of Eq.(3.17) below in order to simplify their calculation.

We note $(\underline{\underline{J}}^{mk})' = \underline{\underline{J}}^{km}$ so that the element a_{mk} of inertia matrix possesses symmetrisation, $a_{mk}=a_{km}$. Thus the calculation of a_{mk} is necessary only for $m \le k$. So the Eqs.(a) and (b) can be written as:

$$a_{mk} = \bar{z}_m \cdot \underline{\underline{J}}^{mk} \cdot \bar{z}_k S'_k S'_m + \sum_{i=k}^{n} [m_i(\bar{z}_k \times \bar{d}_{ki}) \cdot \bar{z}_m S'_k S_m + m_i \bar{z}_k \cdot \bar{z}_m S_k S_m$$
$$+m_i(\bar{z}_m \times \bar{d}_{mi}) \cdot \bar{z}_k S_k S'_m] \qquad (k \ge m)$$

$$\underline{\underline{J}}^{mk} = \sum_{i=k}^{n} [\underline{\underline{J}}_i - m_i(\hat{\underline{\underline{d}}}_{mi} \cdot \hat{\underline{\underline{d}}}_{ki})] \qquad (k \ge m)$$

It is well known that the element b_{mkj} of centrifugal and coriolis effect matrix possesses the following properties:

(1) the marks k and j possess symmetrisation: $b_{mkj}=b_{mjk}$;

(2) the marks m and k possess skew-symmetrisation: $b_{mkj}=-b_{kmj}$ $j \le k,m$;

(3) $b_{kkj}=0 \quad j\leq k$.

Because of these properties, It is sufficient to compute b_{mkj} for $k\geq j$ and $k>m$. In this case, i=max(m,k,j)=k in Eqs.(d), (e) and (g).

Thus, the number of a_{mk} which have to be calculated is reduced from n^2 to $\frac{n^2}{2}+\frac{n}{2}=\frac{1}{2}n(n+1)$ by making use of the symmetrisation of a_{mk}. The number of b_{mkj} which have to be calculated is reduced from n^2 to $\frac{1}{3}n(n^2-1)$ by making use of symmetrisation and skew-symmetrisation of b_{mkj}.

By making use of the symmetrisation of b_{mkj}, the second item in the Eq.(3.17) can be written as:

$$\sum_{k=1}^{n}\sum_{j=1}^{n} b_{mkj}\dot{q}_j\dot{q}_k = 2\sum_{k=1}^{n}\sum_{j=1}^{k-1} b_{mkj}\dot{q}_j\dot{q}_k + \sum_{k=1}^{n} b_{mkj}\dot{q}_k^2$$

where,

$b_{mkj}=H_{mkj}+D_{mkj}$; $b_{mkk}=C_{mkk}+D_{mkk}$.

3.7 Computation Formulae of the Coefficients of the Dynamical Equations

In the following we discuss the coefficients in the Eq.(3.17) according to the types of joints. The computational formulae in different cases can be obtained ($k\geq m$).

(1) Inertia matrix elements a_{mk}:

when $S_m=S_k=0$: $\quad a_{mk}=\bar{z}_m\cdot\underline{\underline{J}}^{mk}\cdot\bar{z}_k$

when $S_m=S_k=1$: $\quad M^k\bar{z}_m\cdot\bar{z}_k$

when $S_m=0$, $S_k=1$: $\quad a_{mk}=\sum_{i=k}^{n} m_i\bar{c}_{ki}\cdot\bar{z}_m$

when S_m=1, S_k=0: $a_{mk} = (\bar{z}_k \times \bar{U}^k) \cdot \bar{z}_m$

(2) Centrifugal and coriolis effect matrix elements b_{mkj}: (k ≥j; k>m)

when $S_m = S_k = S_j = 0$:

$$H_{mkj} = \bar{z}_m \cdot \underline{\underline{J}}^{mk} \cdot (\bar{z}_j \times \bar{z}_k);$$

$$C_{mkk} = \bar{z}_k \times (\bar{z}_k \times \bar{U}^k) \cdot \sum_{i=k}^{n} \bar{c}_{mi};$$

$$D_{mkj} = \sum_{i=k}^{n} \bar{z}_m \cdot \bar{z}_j \times (\underline{\underline{J}}_i \cdot \bar{z}_k);$$

when $S_m = S_k = S_j = 1$:

$$H_{mkj} = 0;$$
$$C_{mkk} = 0;$$
$$D_{mkj} = 0.$$

when S_m=1; $S_k = S_j$=0:

$$H_{mkj} = \bar{z}_m \cdot [(\bar{z}_j \times \bar{z}_k) \times \bar{U}^k];$$
$$C_{mkk} = \bar{z}_m \cdot [\bar{z}_k \times (\bar{z}_k \times \bar{U}^k)];$$
$$D_{mkj} = 0.$$

when $S_m = S_k$=1; S_j=0:

$$H_{mkj} = \bar{z}_m \cdot (\bar{z}_j \times \bar{z}_k) M^k;$$
$$C_{mkk} = 0;$$
$$D_{mkj} = 0.$$

when $S_m = S_j$=1; S_k=0:

$$H_{mkj} = 0;$$
$$C_{mkk} = 0;$$
$$D_{mkj} = 0.$$

when S_m=0; $S_k = S_j$=1:

$$H_{mkj} = 0;$$
$$C_{mkk} = 0;$$
$$D_{mkj} = 0.$$

when $S_m = S_k$=0; S_j=1:

$$H_{mkj} = 0;$$
$$C_{mkk} = 0;$$
$$D_{mkj} = 0.$$

when $S_m = S_j$=0; S_k=1:

$$H_{mkj} = (\bar{z}_j \times \bar{z}_k) \cdot \sum_{i=k}^{n} \bar{c}_{mi} m_i;$$
$$C_{mkk} = 0;$$
$$D_{mkj} = 0.$$

(3) Gravity effects column matrix elements E_m:

when S_m=0: $$-E_m = \bar{z}_m \cdot \sum_{i=m+1}^{n} (\bar{d}_{ii} - \bar{d}_{i,i-1}) \times \bar{z}_i \tau_i S_i + \sum_{i=m}^{n} \bar{c}_{mi} \cdot (\tau_i S_i \bar{z}_i - \tau_{i+1} S_{i+1} \bar{z}_{i+1}) + (\bar{z}_m \times \overline{U}^m) \cdot \bar{g}$$

when S_m=1: $$-E_m = M^m \bar{z}_m \cdot \bar{g}$$

Up to now, the computational formulae for coefficients of Eq.(3.17) in different cases have all been obtained, which make the modelling procedure simple and clear. It is suitable for computer programming.

In the discussion, it is clear that the computational formulae are the most complicated and the amount of calculation is the largest when all of the joints of manipulators are revolute joints. If a prismatic joint exists in the system, the amount of calculation can be reduced greatly.

CHAPTER 4

AN EFFICIENT RECURSIVE APPROACH FOR COMPUTER GENERATION OF MANIPULATOR DYNAMIC MODEL

4.1 Introduction

The dynamic characteristics of manipulators are highly nonlinear and coupling. To control these system, it is necessary to search for a computer-oriented algorithm that fast forms and solves dynamical equations automatically. This necessitates the establishment of a satisfactory manipulator dynamical algorithm. Researchers in the past have applied simplifying assumptions to the model for the manipulator under consideration so that a solution could be obtained. These assumptions have ignored coriolis effects and considered some system elements to be massless . The dynamic equations thus obtained may only be valid in a limited range. Recently, more general dynamic equations have been obtained in which most of the simplifying assumptions have been removed. The present emphasis is specifically placed on computational efficiency. The Largrange formulation (recursive and nonrecursive) and the recursive Newton-Euler formulation of manipulator dynamics have been used as the bases for algorithms that predict generalised actuator forces [18, 20, 27, 30-31]. These algorithms, typically numerical in nature, require many matrix multiplications and vector operations on arrays of floating point numbers. The speed of computation is not fast enough for on-line control of manipulator dynamics using a microcomputer. However, deriving equations using this approach is substantially more complex and the derived equations are difficult to generalise for an arbitrary manipulator geometry.

This chapter presents a algorithm that is a computer-oriented algorithm that forms and solves dynamical equations automatically and is able to compute the dynamic model which minimises the number of arithmetic operations on the basis of the general dynamical equations presented in chapter 3. The algorithm used the augmented body and composite system for the serial robot arm in the dynamical model. The recursive equations corresponding to the augmented body and composite systems are set up. The dynamical algorithm is established by the computer automatically through the recursive equations, which is based only on the geometry of the system.

In order to simplify the presentation, the worst case of robot arm with n joints of the revolute type is studied, as it is well known that the dynamic model is simpler when the manipulator possesses some joints of the prismatic type. But the method presented in this chapter is general and can be applied to an arbitrary manipulator with revolute and/or prismatic joints.

4.2 Robot Arm Dynamic Model

We discuss the serial manipulator system that consists of the n+1 links with revolute joints. Let the type indicative symbol S_i.=0, the dynamically model for the manipulator with the revolute joints can be obtained from the general dynamically model presented in chapter 3 as follows:

1) The angular and linear velocity of link i:

$$\overline{\omega}_i = \sum_{k=1}^{i} \overline{z}_k \dot{q}_k \tag{4.1}$$

$$\overline{v}_i = \sum_{k=1}^{i} (\overline{z}_k \times \overline{d}_{ki}) \dot{q}_k \tag{4.2}$$

2) The angular and linear accelerations of link i

$$\dot{\bar{\omega}}_i = \sum_{k=1}^{i} \bar{z}_k \ddot{q}_k + \sum_{k=1}^{i} \sum_{j=1}^{k-1} (\bar{z}_j \times \bar{z}_k) \dot{q}_k \dot{q}_j \tag{4.3}$$

$$\begin{aligned} \dot{\bar{v}}_i = & \sum_{k=1}^{i} (\bar{z}_k \times \bar{d}_{ki}) \ddot{q} + \sum_{k=1}^{i} \sum_{j=1}^{k-1} [(\bar{z}_j \times \bar{z}_k) \times \bar{d}_{kj}] \\ & + \sum_{k=1}^{i} \sum_{j=k}^{i} [\bar{z}_k \times (\bar{z}_j \times \bar{d}_{ji})] \dot{q}_j \dot{q}_k \end{aligned} \tag{4.4}$$

3) The dynamical equations of the manipulator:

$$\tau_m = \sum_{k=1}^{n} a_{mk} \ddot{q}_k + \sum_{k=1}^{n} \sum_{j=1}^{n} b_{mkj} \dot{q}_j \dot{q}_k + E_m \qquad (m=1,2,.....n) \tag{4.5}$$

where:

$$a_{mk} = \bar{z}_m \cdot \underline{\underline{J}}^{mk} \cdot \bar{z}_k \tag{4.6}$$

$$b_{mkj} = \begin{cases} H_{mkj} + D_{mkj} & j < k \\ C_{mkj} + D_{mkj} & j \geq k \end{cases} \tag{4.7}$$

$$H_{mkj} = \bar{z}_m \cdot \underline{\underline{J}}^{mk} \cdot (\bar{z}_j \times \bar{z}_k) \tag{4.8}$$

$$C_{mkj} = \sum_{i=max(m,k,j)}^{n} (\bar{z}_k \times \bar{c}_{ji}) \cdot \bar{c}_{mi} m_i \tag{4.9}$$

$$D_{mkj} = \sum_{i=max(m,k,j)}^{n} \bar{z}_m \cdot \bar{z}_j \times (\underline{\underline{J}}_i \cdot \bar{z}_k) \tag{4.10}$$

$$E_m = g\bar{z}_1 \cdot (\bar{z}_m \times \bar{U}^m) \tag{4.11}$$

$$\underline{\underline{J}}^{mk} = \sum_{i=max(m,k)}^{n} [\underline{\underline{J}}_i - m_i (\hat{\underline{\underline{d}}}_{mi} \cdot \hat{\underline{\underline{d}}}_{ki})] \tag{4.12}$$

$$\bar{c}_{mi} = \bar{z}_m \times \bar{d}_{mi} \tag{4.13}$$

$$\bar{U}^m = \sum_{i=m}^{n} m_i \bar{d}_{mi} \tag{4.14}$$

The calculation of coefficients of the Eqs.(4.5) are reduced to that of $\underline{\underline{J}}^{mk}$, $\bar{c}_{mi}$ and $\overline{U}^{m}$ using equations (4.6) through (4.14). If it is noted that $(\underline{\underline{J}}^{mk}) = \underline{\underline{J}}^{km}$, then only calculation of $\underline{\underline{J}}^{mk}$ is necessary for m ≤ k. Thus equation (4.12) can be written as:

$$\underline{\underline{J}}^{mk} = \sum_{i=k}^{n} [\underline{\underline{J}}_i - m_i(\hat{\underline{d}}_{mi} \cdot \hat{\underline{d}}_{mk})] \qquad \text{(k≥m)} \tag{4.15}$$

It is well known that the b_{mkj} possess the following properties:

(1) $b_{mkj} = b_{mjk}$; (4.16)

(2) $b_{mkj} = -b_{kmj}$ j≤k,m ; (4.17)

(3) $b_{mmj} = 0$ j≤m . (4.18)

It is sufficient to compute b_{mkj} for k ≥ j and k > m. This makes the number of these coefficients b_{mkj} which have to be calculated equal to (n-1)n(n+1)/3. Then second term in the equation (4.5) can be written as:

$$\sum_{k=1}^{n}\sum_{j=1}^{n} b_{mkj}\dot{q}_j\dot{q}_k = 2\sum_{k=1}^{n}\sum_{j=1}^{k-1} b_{mkj}\dot{q}_j\dot{q}_k + \sum_{k=1}^{n} b_{mkj}\dot{q}_k^2 \tag{4.19}$$

The equations (4.9) and (4.10) can be written as:

$$C_{mkk} = \bar{z}_k \times (\bar{z}_k \times \overline{U}^{k}) \cdot \sum_{i=k}^{n} \bar{c}_{mi} \qquad \text{k > m} \tag{4.20}$$

$$D_{mkj} = \sum_{i=k}^{n} [\bar{z}_m \cdot \bar{z}_j \times (\underline{\underline{J}}_i \cdot \bar{z}_k)] \qquad \text{k > m,j} \tag{4.21}$$

In the following we give the recursive relations of $\underline{\underline{J}}^{mk}$, $\bar{c}_{mi}$ and $\overline{U}^{m}$.

4.3 Augmented Body And Composite System

Augmented body i is defined as the fictitious body composed of link i and the mass of links i+1 through n attached at point O_{i+1} (Fig. 4.1) [39].

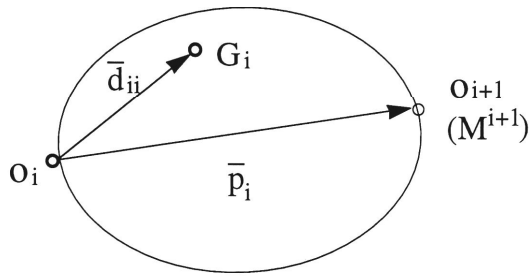

Fig. 4.1 The augmented body

According to the definition of augmented body we have:

$$M^i = \sum_{k=i}^{n} m_k \tag{4.22}$$

$$\bar{u}^i = m_i \bar{d}_{ii} + M^{i+1} + \bar{p}_i \tag{4.23}$$

$$\underline{\underline{k}}^i = \underline{\underline{J}}_i - m_i \underline{\underline{\hat{d}}}_{ii} \cdot \underline{\underline{\hat{d}}}_{ii} - M^{i+1} \underline{\underline{\hat{p}}}_i \cdot \underline{\underline{\hat{p}}}_i \tag{4.24}$$

where previously undefined symbols are:

M^i--- the mass of augmented body i;

$\bar{u}^i$--- the static moment of augmented body i with respect to point O_i.

$\underline{\underline{k}}^i$--- the inertia tensor of augmented body i with respect to point O_i.

Composite system i is defined as the real body composed of link i through n (Fig.4.2).

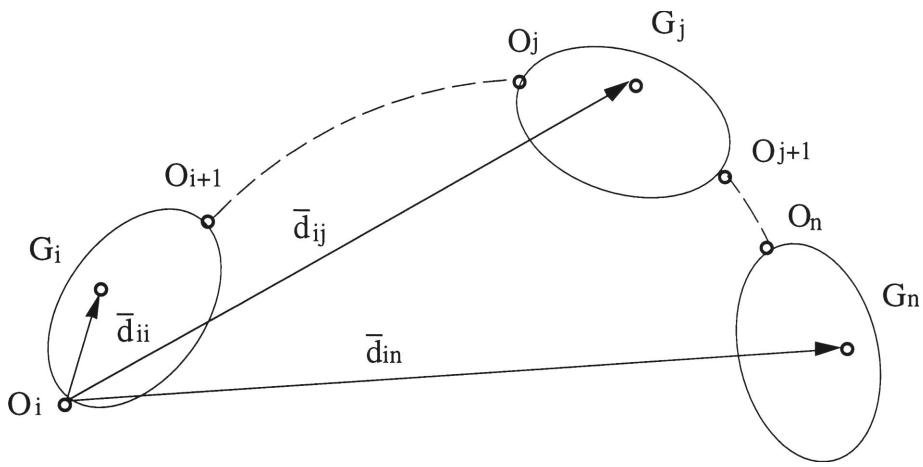

Fig. 4.2 The composite System

According to the definition of composite system, we have:

$$\overline{U}^i = \sum_{k=i}^{n} m_k \overline{d}_{ik} \tag{4.25}$$

$$\underline{\underline{K}}^i = \sum_{k=i}^{n} \underline{\underline{J}}_k - m_k \underline{\underline{\hat{d}}}_{ik} \cdot \underline{\underline{\hat{d}}}_{ik} \tag{4.26}$$

where

$\overline{U}^i$ ---the static moment of composite system i with respect to point O_i.

$\underline{\underline{K}}^i$ ---the inertia tensor of composite system i with respect to point O_i.

From Eqs.(4.22), (4.23), (4.24), (4.25) and (4.26), we obtain the recursive relations:

$$\overline{U}^i = \overline{U}^{i+1} + \overline{u}^i \tag{4.27}$$

$$\underline{\underline{K}}^i = \underline{\underline{K}}^{i+1} + \underline{\underline{k}}^i - \{ \underline{\underline{\hat{p}}}_i, \underline{\underline{\hat{U}}}^{i+1} \} \qquad i = n, n - 1, \ldots \ldots, 1. \tag{4.28}$$

and

$$\overline{d}_{mi} = \overline{d}_{m+1,i} + \overline{p}_m \qquad i = 2, 3, \ldots \ldots, n; \quad m = i - 1, i - 2, \ldots \ldots, 1. \tag{4.29}$$

where

$$\{\hat{\underline{\underline{p}}}_i, \hat{\underline{\underline{U}}}^{i+1}\} = \hat{\underline{\underline{p}}}_i \cdot \hat{\underline{\underline{U}}}^{i+1} + \hat{\underline{\underline{U}}}^{i+1} \cdot \hat{\underline{\underline{p}}}_i \ .$$

From Eqs.(4.15) and (4.26), we derived the following relation:

$$\underline{\underline{J}}^{mk} = \underline{\underline{K}}^{k} - \hat{\underline{\underline{p}}}_{mk} \cdot \hat{\underline{\underline{U}}}^{k} \qquad \text{m} \le \text{k} \tag{4.30}$$

where

$$\overline{p}_{mk} = \sum_{i=m}^{k-1} \overline{p}_i$$

Hence, Eqs.(4.6) and (4.8) become:

$$\begin{cases} a_{mm} = \overline{z}_m \cdot \underline{\underline{K}}^{m} \cdot \overline{z}_m \\ a_{mk} = \overline{z}_m \cdot \underline{\underline{K}}^{k} \cdot \overline{z}_k + (\overline{z}_k \times \overline{U}^{k}) \sum_{i=m}^{k-1} (\overline{z}_m \times \overline{p}_i) \end{cases} \tag{4.31}$$

$$\begin{cases} H_{mmj} = \overline{z}_m \cdot \underline{\underline{K}}^{m} \cdot (\overline{z}_j \times \overline{z}_m) \\ H_{mkj} = \overline{z}_m \cdot \underline{\underline{K}}^{k} \cdot (\overline{z}_j \times \overline{z}_k) + [(\overline{z}_j \times \overline{z}_k) \times \overline{U}^{k}] \sum_{j=m}^{k-1} (\overline{z}_m \times \overline{p}_i) \end{cases} \tag{4.32}$$

$\overline{U}^{i}$, $\underline{\underline{K}}^{i}$ and $\overline{d}_{mi}$ are computed iteratively from the recursive equations (4.27), (4.28) and (4.29). Thus, the calculation of the coefficients of dynamic equations (4.5) needs only to know the constant parameters $\overline{z}_i$, $\overline{d}_{ii}$, $\overline{p}_i$ and $\underline{\underline{J}}_i$ in their own coordinator systems.

4.4 Compact Recursive Procedure

Because the vectors attached to the body are of constant quantity in their own coordinate systems, the recursive calculations are carried out on the associated link coordinate system.

The procedure is:

STEP 1: $\underline{\underline{K}}^{i}_{(i+1)} = \underline{\underline{K}}^{i+1}_{(i+1)} + \underline{\underline{k}}^{i}_{(i+1)} - \{ \underline{\underline{\hat{p}}}_{i}, \underline{\underline{\hat{U}}}^{i+1} \}_{(i+1)}$

$$\overline{U}^{i}_{(i+1)} = \overline{U}^{i+1}_{(i+1)} + \overline{u}^{i}_{(i+1)}$$

STEP 2: $\underline{\underline{K}}^{i}_{(i)} = A_{i,i+1} \cdot \underline{\underline{K}}^{i}_{(i+1)} \cdot A_{i+1,i}$

$$\overline{U}^{i}_{(i)} = A_{i,i+1} \overline{U}^{i}_{(i+1)}$$

STEP 3: $\overline{d}_{mi(m+1)} = \overline{d}_{m+1,i(m+1)} + \overline{p}_{m(m+1)}$

$$\overline{d}_{mi(m)} = A_{m,m+1} \cdot \overline{d}_{mi(m+1)}$$

Step 1 and step 2 are repeated for i = n, n - 1, ..., 1 with the initial conditions:

$$\underline{\underline{K}}^{n}_{(n+1)} = \underline{\underline{k}}^{n}_{(n+1)}; \quad \overline{U}^{n}_{(n+1)} = \overline{u}^{n}_{(n+1)}$$

Step 3 is repeated for i = 2, 3, ..., n and m = i - 1, i - 2, ..., 1 with the initial conditions $\overline{d}_{ii}$ and $\overline{p}_{i-1}$. $A_{i,i+1}$ is the transformation matrix between coordinate systems $(\overline{X}_i, \overline{Y}_i, \overline{Z}_i)$ and $(\overline{X}_{i+1}, \overline{Y}_{i+1}, \overline{Z}_{i+1})$. The terms $\underline{\underline{k}}^{i}_{(i+1)}$ and $\overline{u}^{i}_{(i+1)}$ are constant quantities which can be previously calculated. The signs in brackets express the corresponding coordinate system.

4.5 Example

In Fig.4.3 the PUMA arm with 6 revolute joint is shown with coordinate systems attached to the links. The parameters shown in Table 4.1.

Table--4.1 Link Parameters for PUMA Arm

Joint	1	2	3	4	5	6
α_i	-90	0	-90	-90	-90	0
a_i	0	a_2	a_3	0	0	0
r_i	0	0	d_3	d_4	0	0
$\bar{x}^i$	0	$\bar{x}^2$	0	0	0	0
$\bar{y}^i$	0	0	0	$\bar{y}^4$	0	0
$\bar{z}^i$	z^1	z^2	z^3	0	0	z^6

Where α_i, a_i and r_i are the link parameters defined by the Denavit-Hartenberg notation [40], and $\bar{x}^i$, $\bar{y}^i$ and $\bar{z}^i$ are the coordinates of mass centre of link i.

The number of multiplications and additions necessary for obtaining the dynamic model are shown in Table 4.2.

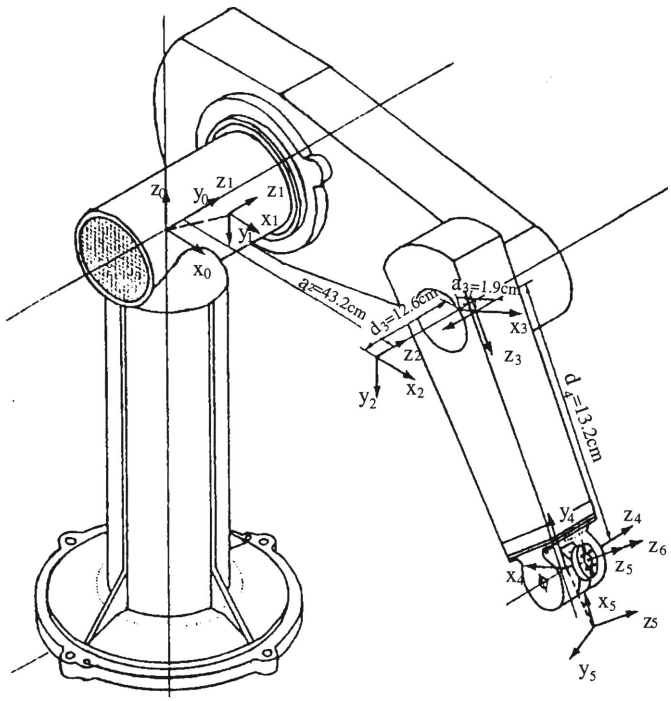

Fig. 4.3 PUMA arm (Unimate 600 robot)

Table--4.2 Number of Computations for the Dynamic Model

Calculation	Mul.	Add.
$\underline{\underline{K}}^i_{(i)}, \overline{U}^i_{(i)}$	52	50
a_{ij}	24	15
b_{ijk}	246	123
E_i	5	2
τ_i	137	129
Total	464	319

4.6 Concluding Remarks

The recursive dynamic modelling method in this chapter is the most simple and direct one in this kind of dynamic modelling method. Although the form of manipulator dynamic equation (4.5) has been discussed by various authors [18, 20, 27, 29-31, 38, 41-46], the method of systematic computation for establishing the expression of manipulator dynamics is new and very efficient. It doesn't require manual derivation of the dynamic equations. The dynamic model can be obtained automatically for any given manipulator using this procedure.

At present, the calculation of the generalised forces has various methods. For a general 6 revolute joint manipulator, the recursive Lagrangian method [30] requires 2195 multiplications and 1719 additions, the recursive Newton-Euler method [18] requires 800 multiplications and 595 additions. The recursive dynamic modelling method in this paper requires 464 multiplications and 319 additions for a particular 6 revolute joints manipulator. It is very efficient to obtain the manipulator dynamic model as demonstrated by the above example. The minimisation of the number of multiplications and additions is due to the fact that some constant quantities can be previously calculated. This makes calculation more efficiency.

CHAPTER 5

DYNAMICS SIMULATION OF MANIPULATORS

5.1 Introduction

Dynamic simulations of the manipulators are investigated in this chapter. In general, the reason for dynamic simulations is to study a system before it is actually built. For that purpose, the model should be as accurate and detailed as possible to closely represent the actual system, so that the predicted behaviour will be close to the actual behaviour of the real system. By simulating dynamic equations, further insight on the dynamic equations of the manipulators can be created.

An industrial robot can be defined, using automatic control terminology, as a nonlinear, coupled, multiple-input/multiple-output system (MIMO). A majority of industrial robots (IR) are driven by dc servomotors. In that case, the inputs are the armature voltages of the motors driving the joints and the outputs are the positions of the joints. Coupling results from the undesired motion of one joint when another joint is driven. Nonlinearities result from the effects of gravitational loading, coriolis and centrifugal torques, friction, etc. Joint control produces the desired position and orientation of the end effector. Solution of dynamical equations to obtain the angle of rotation of rotary joints or displacement distance of prismatic joints as functions of end effector position and orientation is commonly referred to as the "inverse dynamical problem". The control laws that govern system behaviour are of utmost importance since they not only determine the accuracy of the desired motion, but also the speed with which a desired task is performed.

Computer simulation of a manipulator provides insight into manipulator operation and control law effectiveness. Many IR's are controlled assuming a linearized, decoupled model. For example, the PUMA 560 employs a PID controller for each joint with the gains established using a linear model and then tuned empirically. However, it has been noted that this arm experiences significant vibrations at low speeds. This indicates the deficiency of the simplified model and control strategy.

A flowchart showing the component parts of a typical dynamic computer simulation for control law development is given in Fig. 5.1. As trained or commanded by a human operator or as automatically generated through programmed algorithms, the complete trajectory is furnished by the Motion Design section. Through a control law, torques are applied by the actuators to the mechanism. Through solution of the dynamic equations, the joint accelerations may be obtained, and through integration, the actual trajectory is determined.

Advanced control strategies require the inclusion of the dynamical model of the manipulators in the control law. However, the dynamics consists of a highly coupled and non-linear set of equations. Thus, this complexity has always presented a major obstacle in real-time dynamic control applications. The computationally efficient solution of this problem will lead to a better comprehension of the key factors effecting robot operations.

Present day manipulators are generally implemented by simple and well defined (PID) controllers. However, to allow these robots to operate under varying conditions, advanced control algorithms are needed to counteract the different changes and allow for wider diversity. This necessitates the inclusion of the system (robot) dynamics in the controller design.

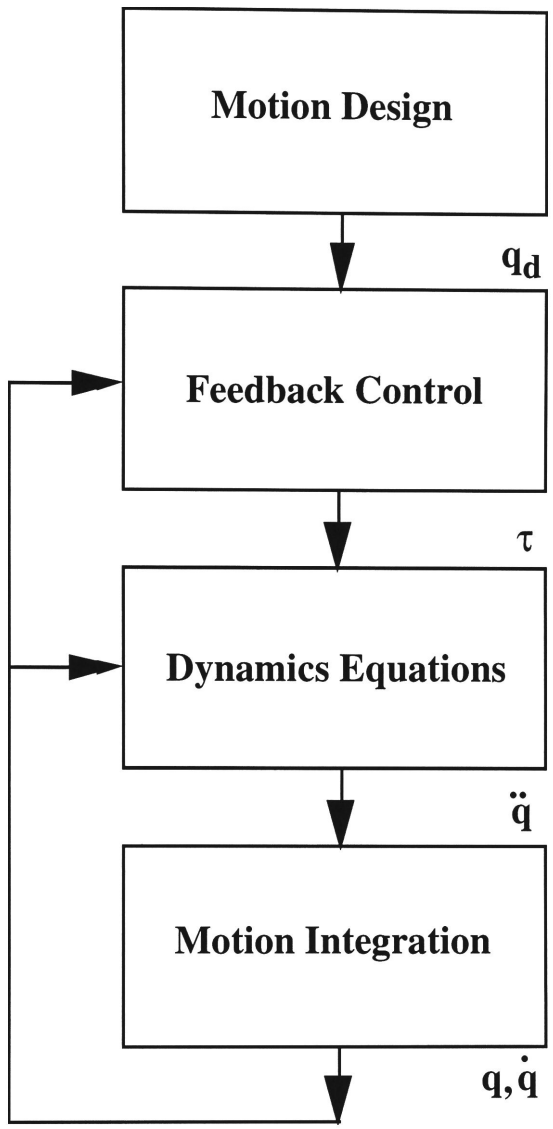

Fig. 5.1 The flowchart of a typical computer simulation for control law development (q, $\dot{q}$, $\ddot{q}$ -- relative joint position, velocity, and acceleration; τ --torque; d -- desired values).

Nevertheless, the computation of the dynamics is a very intensive task. In addition, it must be computed within a sampling rate of 60 Hz or more to avoid poor performance. This emphasises the fact that the efficient and inexpensive computation of the dynamics enhances the feasibility of real-time robot controllers.

The dynamics provide an important tool for simulation and feedforward computations. Therefore, it can be used in testing and designing controllers without the expense and hazards accompanying with actual systems.

The dynamics of manipulators can be categorised in two parts: the inverse dynamics and the direct dynamics. Many modern control schemes for manipulators require the inverse dynamics that determine the actuator forces for the prescribed joint displacements, velocities and accelerations.

The direct dynamics are required while the motion simulation of manipulators is performed. The object is to solve the joint accelerations when the actuator forces are given as input values. The joint velocities and displacements can be obtained by integrating the joint accelerations, and are then used to compute the bias vector. The research of the forward dynamics focuses on the formulation of the inertia matrix and the bias vector.

This chapter is organised as follows. In section 5.2, the computational formulae are derived. In section 5.3, the direct dynamical algorithm of the system based on our modelling method is presented. Then, the inverse dynamical algorithm is developed in section 5.4. Finally, case study of the PUMA 560 robot is presented in section 5.5.

5.2 Computational Formulae

Based on the dynamical model and the computational procedure described in the Chapter 4, the following computational formulae can be derived.

$$\overline{p}_i = \gamma_i \overline{z}_i + a_i \overline{x}_{i+1}$$

$$\begin{cases} \overline{p}_{i(i+1)} = \begin{pmatrix} a_i & \gamma_i \sin\alpha_i & \gamma_i \cos\alpha_i \end{pmatrix}^T = \begin{pmatrix} p_{ix} & p_{iy} & p_{iz} \end{pmatrix}^T_{(i+1)} \\ \overline{d}_{ii(i+1)} = \overline{p}_{i(i+1)} + \overline{G}_{i(i+1)} = \begin{pmatrix} d_{ix} & d_{iy} & d_{iz} \end{pmatrix}^T_{(i+1)} \\ \overline{u}^i_{(i+1)} = M^{i+1}\overline{p}_{i(i+1)} + m_i \overline{d}_{ii(i+1)} = \begin{pmatrix} u_{ix} & u_{iy} & u_{iz} \end{pmatrix}^T_{(i+1)} \\ \underline{\underline{k}}^i_{(i+1)} = \underline{\underline{J}}_i - m_i \hat{\underline{\underline{d}}}_{ii} \cdot \hat{\underline{\underline{d}}}_{ii} - M^{i+1} \hat{\underline{\underline{p}}}_i \cdot \hat{\underline{\underline{p}}}_i \end{cases} \tag{5.1}$$

$$= \underline{\underline{J}}_i - m_i \begin{bmatrix} d_{ix}^2 + d_{iz}^2 & -d_{ix}d_{iy} & -d_{ix}d_{iz} \\ -d_{ix}d_{iy} & d_{iz}^2 + d_{ix}^2 & -d_{iy}d_{iz} \\ -d_{ix}d_{iz} & -d_{iy}d_{iz} & d_{ix}^2 + d_{iy}^2 \end{bmatrix}_{(i+1)}$$

$$+ M^{i+1} \begin{bmatrix} p_{iy}^2 + p_{iz}^2 & -p_{ix}p_{iy} & -p_{ix}p_{iz} \\ -p_{ix}p_{iy} & p_{iz}^2 + p_{ix}^2 & -p_{iy}p_{iz} \\ -p_{ix}p_{iz} & -p_{iy}p_{iz} & p_{ix}^2 + p_{iy}^2 \end{bmatrix}_{(i+1)}$$

$$= \begin{bmatrix} k_{11}^i & k_{12}^i & k_{13}^i \\ k_{12}^i & k_{22}^i & k_{23}^i \\ k_{13}^i & k_{23}^i & k_{33}^i \end{bmatrix}_{(i+1)}$$

where $(p_{ix}, p_{iy}, p_{iz})^T_{(i+1)}, (d_{ix}, d_{iy}, d_{iz})^T_{(i+1)}, (u_{ix}, u_{iy}, u_{iz})^T_{(i+1)}$ are the projective coordinates of $\overline{p}_i, \overline{d}_{ii}, \overline{u}^i$ at the corresponding coordinate axes of the coordinate system i+1, respectively. k^i_{rs} is the element of $\underline{\underline{k}}^i$ at the r-th column and the s-th row.

After computing these constants of $\overline{p}_i, \overline{d}_{ii}, \overline{u}^i, \underline{\underline{k}}^i$, the $\overline{U}^i_{(i+1)}, \underline{\underline{K}}^i_{(i+1)}$ and $\overline{U}^i_{(i)}, \underline{\underline{K}}^i_{(i)}$ can be computed by using the recursive formulae:

$$\begin{cases} \overline{U}^i_{(i+1)} = \begin{pmatrix} U^i_x & U^i_y & U^i_z \end{pmatrix}^T_{(i+1)} \\ \underline{\underline{K}}^i_{(i+1)} = \underline{\underline{K}}^{i+1}_{(i+1)} + \underline{\underline{k}}^i_{(i+1)} - \left\{ \underline{\underline{\hat{p}}}_i, \quad \underline{\underline{\hat{U}}}^{i+1} \right\}_{(i+1)} \end{cases} \tag{5.2}$$

$$= \underline{\underline{K}}^{i+1}_{(i+1)} + \underline{\underline{k}}^i_{(i+1)} +$$

$$+ \begin{bmatrix} 2(p_{iz}U^i_z + p_{iz}U^i_z) & -p_{iy}U^i_x - p_{ix}U^i_y & -p_{iz}U^i_x - p_{ix}U^i_z \\ -p_{iy}U^i_x - p_{ix}U^i_y & 2(p_{ix}U^i_x + p_{iz}U^i_z) & -p_{iy}U^i_z - p_{iz}U^i_y \\ -p_{iz}U^i_x - p_{ix}U^i_z & -p_{iy}U^i_z - p_{iz}U^i_y & 2(p_{iy}U^i_y + p_{iz}U^i_z) \end{bmatrix}_{(i+1)}$$

$$= \begin{bmatrix} K^i_{11} & -K^i_{12} & -K^i_{13} \\ -K^i_{12} & K^i_{22} & -K^i_{23} \\ -K^i_{13} & -K^i_{23} & K^i_{33} \end{bmatrix}_{(i+1)}$$

From the Eqs. (4.27) and (4.28), the follows can be obtained:

$$\begin{cases} \underline{\underline{K}}^i_{(i)} = \begin{bmatrix} K^i_{11} & -K^i_{12} & -K^i_{13} \\ -K^i_{12} & K^i_{22} & -K^i_{23} \\ -K^i_{13} & -K^i_{23} & K^i_{33} \end{bmatrix}_{(i)} \\ \overline{U}^i_{(i)} = \begin{pmatrix} U^i_x & U^i_y & U^i_z \end{pmatrix}^T_{(i)} \end{cases} \tag{5.2'}$$

where, $\begin{pmatrix} U^i_x & U^i_y & U^i_z \end{pmatrix}^T_{(i+1)}, \begin{pmatrix} U^i_x & U^i_y & U^i_z \end{pmatrix}^T_{(i)}$ are the projective coordinates of $\overline{U}^i$ at the corresponding coordinate axes of the coordinate system i+1 and i, respectively. $K^i_{rs,(i+1)}, K^i_{rs,(i)}$ are the elements of $\underline{\underline{K}}^i$ at the r-th column and the s-th row in the coordinate system i+1 and i, respectively.

By using the coordinate transformation to $\overline{p}_i$, $\overline{z}_i$, the following can be derived:

$$\begin{cases} \bar{p}_{i(j)} = \left(p_{ix} \quad p_{iy} \quad p_{iz}\right)^T_{(j)} \\ \bar{z}_{i(j)} = A_{j,i}\bar{z}_{i(i)} = \left(A^{13}_{ji} \quad A^{23}_{ji} \quad A^{33}_{ji}\right)^T \\ (\bar{z}_i \times \bar{p}_l)_{(j)} = \hat{\tilde{z}}_{i(j)} \cdot \bar{p}_{l(j)} = \left(l_{il} \quad m_{il} \quad n_{il}\right)^T_{(j)} \\ (\bar{z}_j \times \bar{U}^j)_{(j)} = \hat{\tilde{z}}_{j(j)} \cdot \bar{U}^j_{(j)} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{U}^j_x \\ \bar{U}^j_y \\ \bar{U}^j_z \end{bmatrix}_{(j)} \\ \qquad = \left(-U^j_y \quad \bar{U}^j_x \quad 0\right)^T \end{cases} \tag{5.3}$$

where, $\left(p_{ix} \quad p_{iy} \quad p_{iz}\right)^T_{(j)}, \left(l_{il} \quad m_{il} \quad n_{il}\right)^T_{(j)}$ are the projective coordinates of $\bar{p}_i$, $\bar{z}_i \times \bar{p}_l$ at the corresponding coordinate axes of the coordinate system j, respectively. A^{rs}_{ji} are the element of A_{ji} at the r-th column and the s-th row.

$$(\bar{z}_i \cdot \underline{\underline{K}}^j \cdot \bar{z}_j)_{(j)} = \bar{z}_{i(j)} \cdot \begin{bmatrix} K^j_{11} & -K^j_{12} & -K^j_{13} \\ -K^j_{12} & K^j_{22} & -K^j_{23} \\ -K^j_{13} & -K^j_{23} & K^j_{33} \end{bmatrix}_{(j)} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= (-A^{13}_{ji}K^j_{13} - A^{23}_{ji}K^j_{23} + A^{33}_{ji}K^j_{33})_{(j)} = E_{ij}$$

$$(\bar{z}_i \cdot \underline{\underline{K}}^i \cdot \bar{z}_i)_{(i)} = \left(0 \quad 0 \quad 1\right) \cdot \underline{\underline{K}}^i_{(i)} \cdot \left(0 \quad 0 \quad 1\right) = K^i_{33,(i)}$$

then, the Eq. (4.31) become:

$$\begin{cases} a_{ii} = K^i_{33,(i)} \\ a_{ij} = E_{ij} + \left(-U^j_y \quad U^j_x \quad 0\right)_{(j)} \sum_{l=i}^{j-1} \begin{bmatrix} l_{il} \\ m_{il} \\ n_{il} \end{bmatrix}_{(j)} \end{cases} \qquad (j \geq i) \tag{5.4}$$

$$(\bar{z}_k \times \bar{z}_i)_{(i)} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} A^{13}_{ik} \\ A^{23}_{ik} \\ A^{33}_{ik} \end{bmatrix} = \left(A^{13}_{ik} \quad -A^{13}_{ik} \quad 0\right)^T \tag{5.5}$$

$$[\bar{z}_i \cdot \underline{\underline{K}}^i \cdot (\bar{z}_k \times \bar{z}_i)]_{(i)} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} K_{11}^i & -K_{12}^i & -K_{13}^i \\ -K_{12}^i & K_{22}^i & -K_{23}^i \\ -K_{13}^i & -K_{23}^i & K_{33}^i \end{bmatrix}_{(i)} \begin{bmatrix} A_{ik}^{23} \\ -A_{ik}^{13} \\ 0 \end{bmatrix}$$

$$= -K_{13}^i A_{ik}^{23} - K_{23}^i A_{ik}^{13} \tag{5.6}$$

Let $[\bar{z}_i \cdot \underline{\underline{K}}^j \cdot (\bar{z}_k \times \bar{z}_j)]_{(j)} = \begin{pmatrix} A_{ji}^{13} & A_{ji}^{23} & A_{ji}^{33} \end{pmatrix} \begin{bmatrix} K_{11}^j & -K_{12}^j & -K_{13}^j \\ -K_{12}^j & K_{22}^j & -K_{23}^j \\ -K_{13}^j & -K_{23}^j & K_{33}^j \end{bmatrix}_{(j)} \cdot$

$$\cdot \begin{bmatrix} A_{jk}^{23} \\ -A_{jk}^{13} \\ 0 \end{bmatrix} = A_{ji}^{13} (K_{11}^j A_{jk}^{23} + K_{12}^j A_{jk}^{13})_{(j)} - A_{ji}^{23} (K_{12}^j A_{jk}^{23} + K_{22}^j A_{jk}^{13})_{(j)}$$

$$-A_{ji}^{33} (K_{13}^j A_{jk}^{23} - K_{23}^j A_{jk}^{13})_{(j)} = Q_{ijk}$$

$$[(\bar{z}_k \times \bar{z}_j) \times \bar{U}^j]_{(j)} = \begin{bmatrix} 0 & 0 & -A_{jk}^{13} \\ 0 & 0 & -A_{jk}^{23} \\ A_{jk}^{13} & A_{jk}^{23} & 0 \end{bmatrix} \begin{bmatrix} \bar{U}_x^j \\ \bar{U}_y^j \\ \bar{U}_z^j \end{bmatrix}_{(j)}$$

$$= \begin{pmatrix} -A_{jk}^{13} \bar{U}_z^j & -A_{jk}^{23} \bar{U}_z^j & A_{jk}^{13} \bar{U}_x^j + A_{jk}^{23} \bar{U}_y^j \end{pmatrix}_{(j)}^T \tag{5.8}$$

then, the Eq.(4.32) becomes:

$$\begin{cases} H_{iik} = -K_{13}^i A_{ik}^{23} - K_{23}^i A_{ik}^{13} \\ H_{ijk} = Q_{ijk} + \begin{pmatrix} -A_{jk}^{13} \bar{U}_z^j & -A_{jk}^{23} \bar{U}_z^j & A_{jk}^{13} \bar{U}_x^j + A_{jk}^{23} \bar{U}_y^j \end{pmatrix}_{(j)} \end{cases} \tag{5.9}$$

$$\cdot \sum_{l=i}^{j-1} \begin{bmatrix} l_{il} \\ m_{il} \\ n_{il} \end{bmatrix}_{(j)} \qquad (k < j, j < i)$$

From the recursive formulae (4.29), the follows can be obtained:

$$\bar{d}_{mi(m)} = \begin{pmatrix} d_{mix} & d_{miy} & d_{miz} \end{pmatrix}_{(m)}^T$$

where, $\begin{pmatrix} d_{mix} & d_{miy} & d_{miz} \end{pmatrix}^T_{(m)}$ are the projective coordinates of $\bar{d}_{mi}$ at the corresponding coordinate axes of the coordinate system m, respectively.

$$\bar{c}_{mi(m)} = (\bar{z}_m \times \bar{d}_{mi})_{(m)} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_{mix} \\ d_{miy} \\ d_{miz} \end{bmatrix}_{(m)}$$

$$= \begin{pmatrix} -d_{miy} & d_{mix} & 0 \end{pmatrix}^T_{(m)} \tag{5.10}$$

By transforming $\bar{c}_{ki(k)}$ into the coordinate system m, we can obtain:

$$\bar{c}_{ki(m)} = \begin{pmatrix} c_{kix} & c_{kiy} & c_{kiz} \end{pmatrix}^T_{(m)}$$

where, $\begin{pmatrix} c_{kix} & c_{kiy} & c_{kiz} \end{pmatrix}^T_{(m)}$ are the projective coordinates of $\bar{c}_{ki}$ at the corresponding coordinate axes of the coordinate system m, respectively.

$$(\bar{z}_k \times \bar{c}_{ki})_{(m)} = \begin{bmatrix} 0 & -A^{33}_{mk} & A^{23}_{mk} \\ A^{33}_{mk} & 0 & -A^{13}_{mk} \\ -A^{23}_{mk} & A^{13}_{mk} & 0 \end{bmatrix} \begin{bmatrix} c_{kix} \\ c_{kiy} \\ c_{kiz} \end{bmatrix}_{(m)}$$

$$= \begin{bmatrix} -A^{33}_{mk} c_{kiy} + A^{23}_{mk} c_{kiz} \\ A^{33}_{mk} c_{kix} - A^{13}_{mk} c_{kiz} \\ -A^{23}_{mk} c_{kix} + A^{13}_{mk} c_{kiy} \end{bmatrix}_{(m)}$$

Therefore, the coefficient C_{mkk} of the Eq.(4.5) can be written:

$$C_{mkk} = \sum_{i=m}^{n} [(\bar{z}_k \times \bar{c}_{ki}) \cdot \bar{c}_{mi} m_i]_{(m)}$$

$$= \sum_{i=m}^{n} \{ \begin{bmatrix} -A^{33}_{mk} c_{kiy} + A^{23}_{mk} c_{kiz} \\ A^{33}_{mk} c_{kix} - A^{13}_{mk} c_{kiz} \\ -A^{23}_{mk} c_{kix} + A^{13}_{mk} c_{kiy} \end{bmatrix}_{(m)} \cdot \begin{bmatrix} -d_{miy} \\ d_{mix} \\ 0 \end{bmatrix}_{(m)} m_i \}$$

$$= \sum_{i=m}^{n} [\, d_{miy}(A_{mk}^{33} c_{kiy} - A_{mk}^{23} c_{kiz}) + d_{mix}(A_{mk}^{33} c_{kix} - A_{mk}^{13} c_{kiz}) \,]_{(m)}$$

$$(m \geq k) \tag{5.11}$$

We compute the following first:

$$(\underline{\underline{J}}_i \cdot \bar{z}_k)_{(j)} = \begin{pmatrix} s_{ikx} & s_{iky} & s_{ikz} \end{pmatrix}_{(j)}^{T}$$

where, $\begin{pmatrix} s_{ikx} & s_{iky} & s_{ikz} \end{pmatrix}_{(j)}^{T}$ are the projective coordinates of $(\underline{\underline{J}}_i \cdot \bar{z}_k)$ at the corresponding coordinate axes of the coordinate system j, respectively.

$$[\bar{z}_j \times (\underline{\underline{J}}_i \cdot \bar{z}_k)]_{(j)} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_{ikx} \\ s_{iky} \\ s_{ikz} \end{bmatrix}_{(j)}$$

$$= \begin{pmatrix} -s_{iky} & s_{ikx} & 0 \end{pmatrix}_{(j)}^{T}$$

Therefore, the coefficient D_{mkj} of the Eq.(4.5) can be written:

$$D_{mkj} = \sum_{i=m}^{n} [\, \bar{z}_m \cdot \bar{z}_j \times (\underline{\underline{J}}_i \cdot \bar{z}_k) \,]_{(j)} m_i$$

$$= \sum_{i=m}^{n} m_i \begin{bmatrix} A_{jm}^{13} \\ A_{jm}^{23} \\ A_{jm}^{33} \end{bmatrix}^{T} \begin{bmatrix} -s_{iky} \\ s_{ikx} \\ 0 \end{bmatrix}_{(j)} = \sum_{i=m}^{n} m_i (-A_{jm}^{13} s_{iky} + A_{jm}^{23} s_{ikx})_{(j)}$$

$$(m \geq k,\ j \leq k) \tag{5.12}$$

The coefficient E_n of the Eq.(4.5) can be written:

$$E_m = [g\bar{z}_1 \cdot (\bar{z}_m \times \overline{U}^m)]_{(m)} = g\begin{bmatrix} A_{m1}^{13} \\ A_{m1}^{23} \\ A_{m1}^{33} \end{bmatrix}^T \begin{bmatrix} -U_y^m \\ U_x^m \\ 0 \end{bmatrix}_{(m)}$$

$$= g(-A_{m1}^{13}U_y^m + A_{m1}^{23}U_x^m)_{(m)} \tag{5.13}$$

Up until now, all the coefficient equations in the dynamical model of chapter 4 are expressed as computational formulae. The codes for computer can be programmed from these computational formulae directly.

5.3 Direct Dynamics

In the preceding subsection, we have formulated $a_{mk}, b_{mk}, C_{mkj}, D_{mkj}, E_m, H_{mkj}$ with respect to the corresponding coordinate system. However, the aim to research manipulator dynamics is not only for deriving the dynamical equations for the practical system but also for dealing with the mathematical operation of these equations in order to decide the motion law and the control law of the systems. It is the final aim to program the general programs and to apply it to the practical problems in manufacture and design.

5.3.1 Computational procedure

To the direct dynamical problem, the computational procedure for computer is constructed as follows.

Step 1: To build the database of manipulator dynamics. To input the link parameters a, α, the inertia tensors $\underline{\underline{J}}_i$, body vectors $\overline{p}_i, \overline{d}_{ii}$ described in corresponding body coordinate systems, and the initial condition $q_{i(0)}, \dot{q}_{i(0)}$ for i=1,2,... ..., n.

Step 2: To compute constant $M^i, \overline{u}^i, \underline{\underline{k}}^i$ for i =1, 2, ..., n.

Step 3: To compute $\overline{U}^i, \underline{\underline{K}}^i, \overline{p}_{i(j)}, \overline{d}_{ij(i)}$ by the recursive formulae for i =1, 2,... ..., n.

Step 4: To compute all the coefficients $a_{mk}, b_{mk}, C_{mkj}, D_{mkj}, E_m, H_{mkj}$ of the equation.

Step 5: To compute the driving force τ_i for i =1, 2, ..., n from the given functions $\tau_i = \tau_i(q_i,...,...,q_n,\dot{q}_i,...,...,\dot{q}_n,t)$.

Step 6: To solve the linear equations of $\ddot{q}_i$ by the conjugate gradient method in order to decide the generalised accelerations $\ddot{q}_i$.

Step 7: To take numerical integral to $\ddot{q}_i$ by the Adams estimation-rectification method in order to decide the q_i and $\dot{q}_i$ for next step.

Step 8: To go to step 3.

Repeatedly to compute from step 3 to step 7 at each dispersed point of time, the motion law of q_i can be decide in whole interval of time. On the following the conjugate gradient method and Adams estimation-rectification method used at step 6 and step 7 are introduced briefly.

5.3.2 The conjugate gradient method

The conjugate gradient method is adopted to solve the linear equations of $\ddot{q}_i$. Using this method, we can get the accurate solution with just finite iterations along the optimum conjugate direction by optimum step size. For the nth-order linear equations $A\bar{x} = b$, n-time iterations are needed at the most using this method. It converts the problem of solving linear equations into the extremum problem of multi-element function.

To solve the linear equations:

$$A\bar{x} = b \tag{5.14}$$

Assuming that A is an n-order symmetric square matrix, and $\bar{\alpha}$ is the solution of the linear equations (5.14). If the $\bar{\alpha}$ minimises the quadratic function,

$$F(\bar{x}) = \bar{x}^T A\bar{x} - 2b^T\bar{x} \tag{5.15}$$

this means that to all $\bar{x} \neq \bar{\alpha}$ we have:

$$F(\bar{\alpha}) < F(\bar{x})$$

Therefore, we get the equiscalar surface of $F(\bar{x})$ in n-dimension vector space. The equiscalar surface is a family of ellipsoids in the n-dimension space. The family of ellipsoids centres on the point $\bar{x} = \bar{\alpha}$. Therefore, solving the linear equations (5.14) is converted into solving the common centre of the n-dimension ellipsoid family.

The computational procedure for the conjugate gradient method is written as follows.

First, let a vector $\bar{x}_0$ be the initial vector, then the vector group $\{\bar{r}_i\}, \{\bar{x}_i\}, \{\bar{p}_i\}$ can be obtained in succession by using the following formulae.

$$\begin{cases} \bar{r}_0 = b - A\bar{x}_0 \\ \bar{p}_0 = \bar{r}_0 \\ \bar{\alpha}_i = \bar{p}_i^T \bar{r}_i / \bar{p}_i^T A\bar{p}_i \\ \bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i \\ \bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i A\bar{p}_i \\ \bar{\beta}_i = -\bar{p}_i^T A\bar{r}_{i+1} / \bar{p}_i^T A\bar{p}_i \\ \bar{p}_{i+1} = \bar{r}_{i+1} + \bar{\beta}_i \bar{p}_i \end{cases}$$

where, $\bar{r}_i$ is the complement vector of the vector $\bar{x}_i$, and $\bar{p}_i$ is the direction vector of the step i.

The conjugate gradient method is in fact a kind of direct method. It does not consider the rounding error of the computation, so the accurate solution can be obtained in finite steps. Due to the rounding error, generally the complement vector $\bar{r}_i$ cannot satisfy accurately the orthogonal relative. Generally $\bar{r}_n \neq 0$. The conjugate gradient method possesses the characteristics of the iteration method. If $\bar{r}_n \neq 0$ after computing n steps, the $\bar{x}_n$ can be taken as the initial vector and the computation can be started again. So long as the initial vector is not equal to zero, the successive approximations will close to the accurate solution more and more. The computation can keep going on this way until the solution satisfies the accuracy requirement or the solution cannot be improved further because of the rounding error.

5.3.3 The Adams estimation-rectification method

The Adams method belongs to the multi-step method. It also has the estimation-rectification scheme. The essential thought of the multi-step method is fully to utilise the approximate values got before such as u_0, u_1, ..., u_n, for computing the u_{n+1} in order to prove the accuracy of computational results. The Adams equations

are divided into the explicit formula and the implicit formula. The estimation-rectification scheme means that the initial estimation of the solution is computed by the explicit formula first, then the initial estimation is rectified by the implicit formula.

The problem of solving the velocities and positions from the accelerations is the initial value problem of the ordinary differential equation. The general form of the initial value problem is as follows:

$$\begin{cases} u' = f(t,u), \quad t \in [t_0, T] \\ u(t_0) = u_0 \end{cases}$$

(5.17)

Integrating Eq.(5.17) from t_n to t_{n+1} leads to:

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(t,u(t))dt \tag{5.18}$$

The initial value problem (5.17) of the differential equation is equal to the integral equation (5.18). Obviously, each numerical integral method for the integral equation (5.18) is a kind of numerical solving method for the initial value problem (5.17). We use the interpolation polynomial $F(t)$ with the interpolating nodes t_{n-2}, t_{n-1}, t_n, t_{n+1} to replace $f(t,u(t))$. Therefore, the following can be written:

$$\begin{aligned} F(t) = & \frac{(t-t_{n-1})(t-t_n)(t-t_{n+1})}{(t_{n-2}-t_{n-1})(t_{n-2}-t_n)(t_{n-2}-t_{n+1})} f[t_{n-2}, u(t_{n-2})] \\ & + \frac{(t-t_{n-2})(t-t_n)(t-t_{n+1})}{(t_{n-1}-t_{n-2})(t_{n-1}-t_n)(t_{n-1}-t_{n+1})} f[t_{n-1}, u(t_{n-1})] \\ & + \frac{(t-t_{n-2})(t-t_{n-1})(t-t_{n+1})}{(t_n-t_{n-2})(t_n-t_{n-1})(t_n-t_{n+1})} f[t_n, u(t_n)] \\ & + \frac{(t-t_{n-2})(t-t_{n-1})(t-t_n)}{(t_{n+1}-t_{n-2})(t_{n+1}-t_{n-1})(t_{n+1}-t_n)} f[t_{n+1}, u(t_{n+1})] \end{aligned} \tag{5.19}$$

By substituting $F(t)$ into Eq.(5.18), the Adams implicit formula can be derived

as follows (it is the so-called Adams extrapolation formula):

$$u_{n+1} = u_n + \frac{h}{24}[9f(t_{n+1}, u_{n+1}) + 19f(t_n, u_n) - 5f(t_{n-1}, u_{n-1}) + f(t_{n-2}, u_{n-2})] \tag{5.20}$$

If we use the interpolation polynomial $F(t)$ with the interpolating nodes t_{n-3}, t_{n-2}, t_{n-1}, t_n to replace $f(t, u(t))$ in Eq.(5.18), the Adams explicit formula can be derived similarly as above (this is the so-called Adams extrapolation formula also):

$$u_{n+1} = u_n + \frac{h}{24}[55f(t_n, u_n) - 59f(t_{n-1}, u_{n-1}) + 37f(t_{n-2}, u_{n-2}) - 9f(t_{n-3}, u_{n-3})] \tag{5.21}$$

The local truncation errors of the Adams equations are as follows:

The implicit formula:

$$u(t_{n+1}) - u_{n+1} = -\frac{19}{720}h^5 u_n^{(5)} + \ \ldots\ \ldots \tag{5.22}$$

The explicit formula:

$$u(t_{n+1}) - u_{n+1} = \frac{251}{720}h^5 u_n^{(5)} + \ \ldots\ \ldots \tag{5.23}$$

Both have fourth order accuracy.

We chose the Adams explicit formula as the "estimation" formula, and the Adams implicit formula as the "rectification" formula. The former provides the initial estimated value to make the latter converge fast. The estimation-rectification system can be expressed as follows:

Estimation :

$$u_{n+1} = u_n + \frac{h}{24}[55f(t_n, u_n) - 59f(t_{n-1}, u_{n-1}) + 37f(t_{n-2}, u_{n-2}) - 9f(t_{n-3}, u_{n-3})]$$

Rectification:

$$u_{n+1} = u_n + \frac{h}{24}[9f(t_{n+1}, u_{n+1}) + 19f(t_n, u_n) - 5f(t_{n-1}, u_{n-1}) + f(t_{n-2}, u_{n-2})]$$

Before the computation starts, the fourth order Runge-Kutta formulae are used to compute the first three steps of values of u and f. Then, by using these values as the start value, we can compute next u by the estimation formula, and use the rectification formula to carry out the iterations. The iteration procedure is carried out repeatedly by the estimation formula and the rectification formula until the given value of t is reached.

Obviously, the Adams estimation-rectification method has higher accuracy, and can reduces the error better, and thereby decrease the influence of the accumulative error to computational results. So, it is a kind of advance numerical integral method.

5.3.4 The flowchart of the computation algorithm for direct dynamics

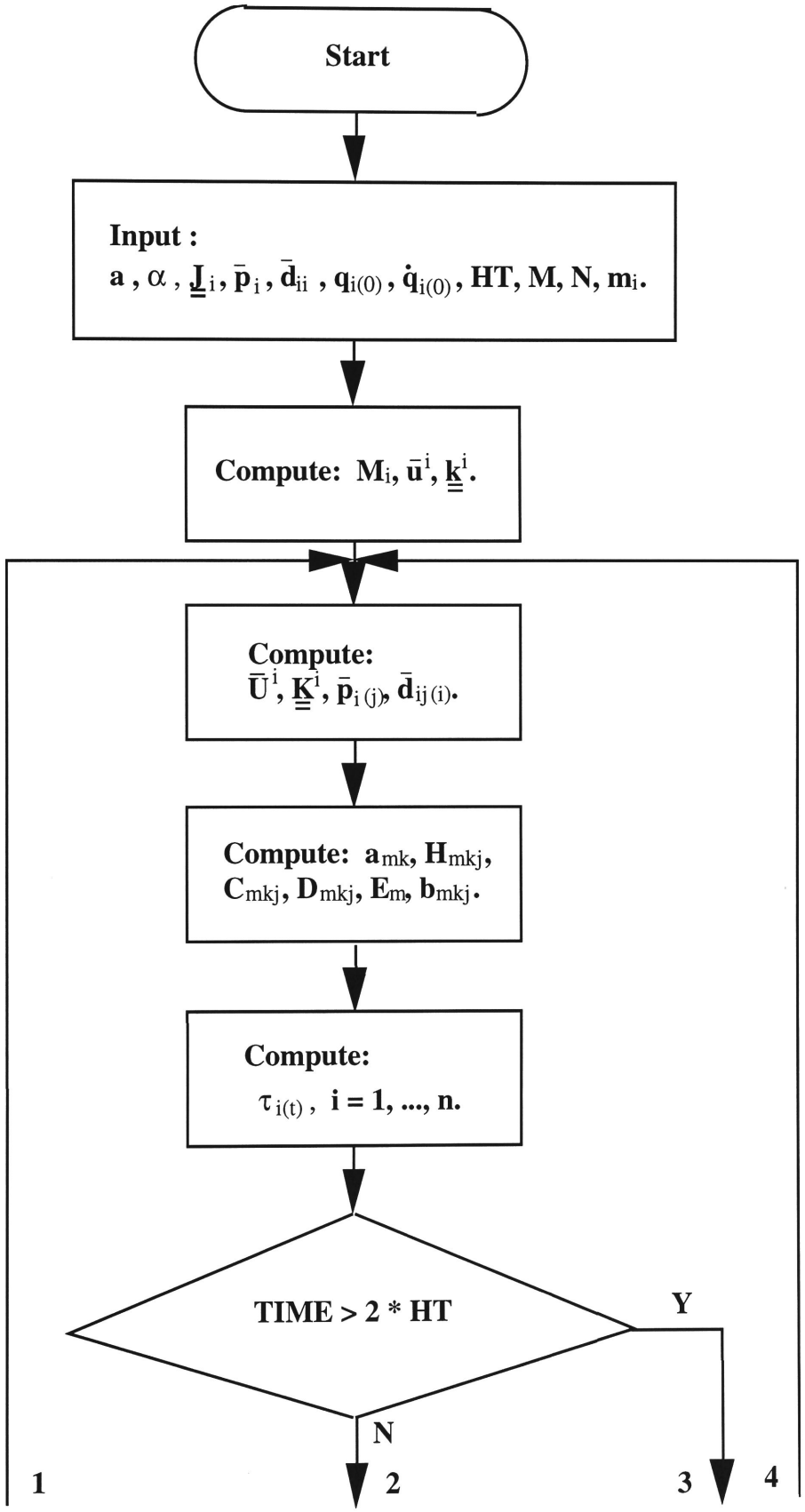

Fig.5.2 The flowchart of the computation algorithm for direct dynamics.

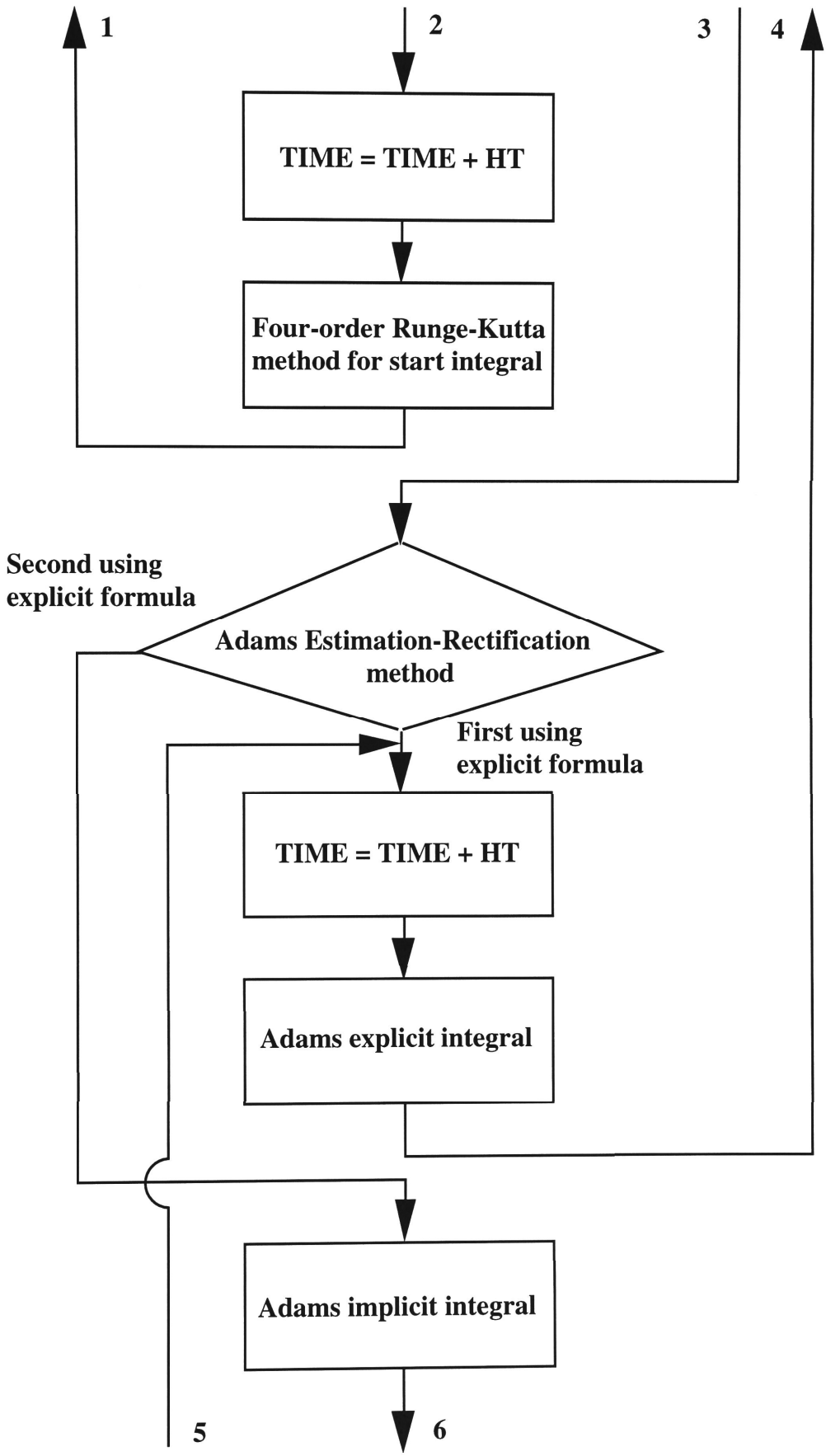

Fig.5.2 The flowchart of the computation algorithm for direct dynamics (continued).

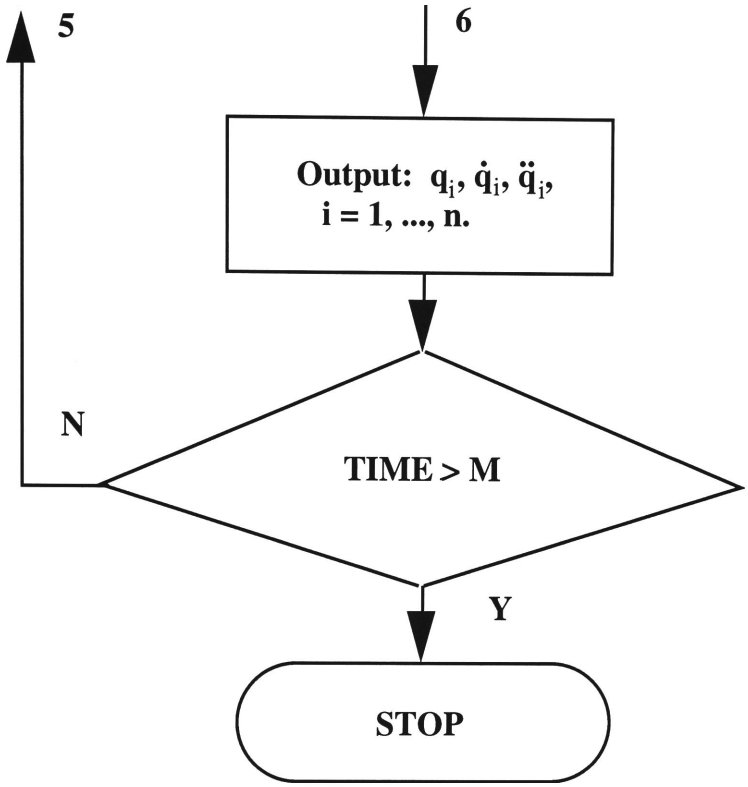

Fig.5.2 The flowchart of the computation algorithm for direct dynamics (continued).

HT --- Step size,

M --- Stop time,

N --- degree of freedom.

The dynamical behaviour is described in terms of the time rate of change of the arm configuration in relation to the joint torques exerted by the actuators. This relationship can be expressed by a set of differential equations that govern the dynamic response of the arm linkage to input joint torques.

5.4 Inverse Dynamics

The so-called robot inverse dynamics is to compute actuator forces or torques (i.e. feedforward control laws) according to the given joint position, velocity, and acceleration. It is very important and useful to real-time robot control and simulation, since this allows one to find the appropriate control input for producing the desired movement. However, from the viewpoint of real-time robot control, developing computationally more simple and efficient approaches for the robot inverse dynamical computation is necessary and significant, since effective real-time control of a robot requires its inverse dynamical computation to be as fast as possible.

5.4.1 Computational procedure

To the direct dynamical problem, the computational procedure for computer is constructed as follows.

Step 1: To build the database of manipulator dynamics. To input the link parameters a, α, the inertia tensors $\underline{\underline{J}}_i$, body vectors $\bar{p}_i, \bar{d}_{ii}$ described in corresponding body coordinate systems for i=1,2,... ..., n.

Step 2: To compute constant $M^i, \bar{u}^i, \underline{\underline{k}}^i$ for i =1, 2, ..., n.

Step 3: To compute $\bar{U}^i, \underline{\underline{K}}^i, \bar{p}_{i(j)}, \bar{d}_{ij(i)}$ by the recursive formulae for i =1, 2,... ..., n.

Step 4: To compute all the coefficients $a_{mk}, b_{mk}, C_{mkj}, D_{mkj}, E_m, H_{mkj}$ of the equation.

Step 5: To compute $q_i(t), \dot{q}_i(t), \ddot{q}_i(t)$ for i = 1, 2, ..., n from the given motion law $q_i = q_i(t)$.

Step 6: To compute the driving force τ_i for i =1, 2, ..., n.

Step 7: To go to step 3.

Repeatedly to compute from step 3 to step 6 at each dispersed point of time, the variety law of τ_i can be decided on whole interval of time. Compared with the computational procedure of direct dynamics, both procedures are essentially the same. But, the procedure of the inverse dynamics is two steps short such as solving the general accelerations by using the conjugate gradient method and taking numerical integral by the Adams estimation-rectification method. This makes the computation of the inverse dynamics simpler than that of direct dynamics. So, the computational time is decreased. Because the essential steps are same, we can develop the general program suitable for the direct and inverse dynamics.

5.4.2 The flowchart of the computation algorithm for inverse dynamics

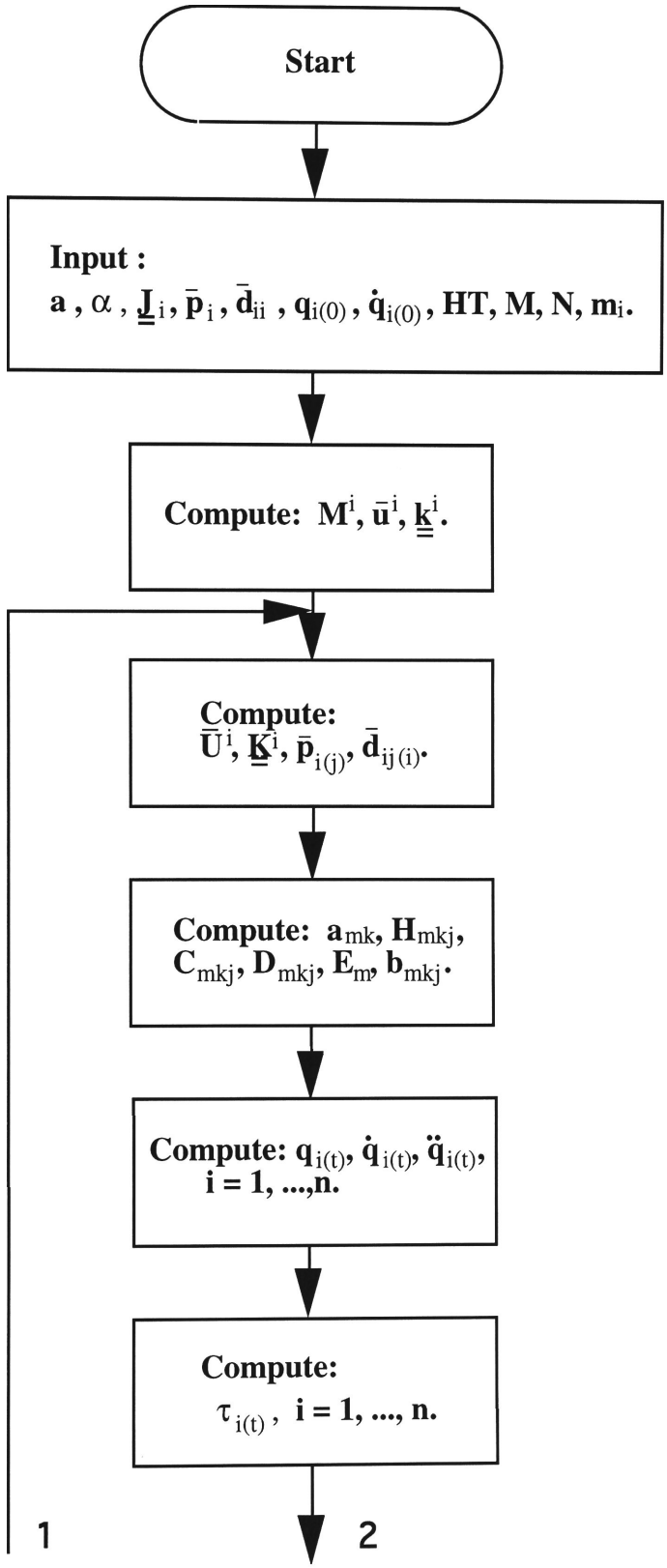

Fig.5.3 The flowchart of the computation algorithm for inverse dynamics.

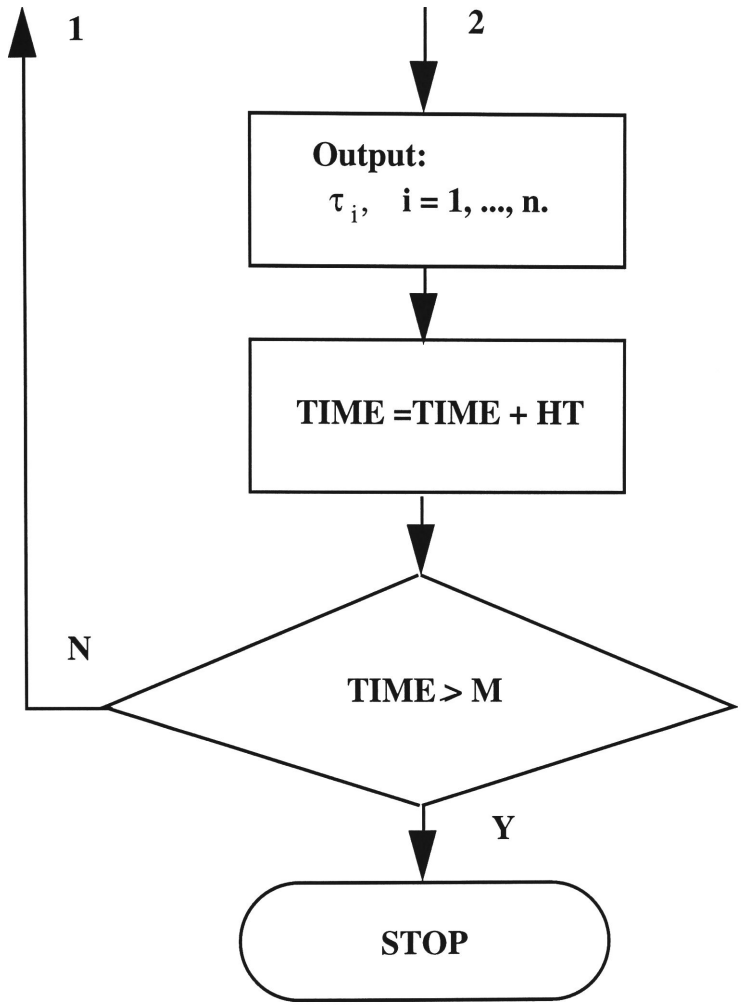

Fig.5.3 The flowchart of the computation algorithm for inverse dynamics (continued).

HT --- Step size,

M --- Stop time,

N --- degree of freedom.

5.5 Example

5.5.1 The database of the PUMA arm

The PUMA arm (Unimate 600 robot) is taken as the example to show the computational results of the direct and inverse dynamics. The construction and the definitions of each coordinate system are shown in the Fig. 4.3. [47]

The construction parameters are shown in the table 5.1, 5.2, 5.3.

Table 5.1 Link parameters

Link	Area (cm^2)	$\overline{X}$ (cm)	$\overline{Y}$ (cm)	$\overline{Z}$ (cm)	m (N)
1	1910	0	0	8	328.3
2	4408	-21.6	0	21.75	757.5
3	2070	0	0	21.6	355.7
4	510	0	2	0	87.7
5	136	0	0	0	23.4
6	57	0	0	1	9.8

Table 5.2 Link parameters (continued)

Joint	1	2	3	4	5	6
α^o	-90	0	-90	-90	-90	0
d (cm)	0	0	12.5	43.2	0	0
a (cm)	0	43.2	1.9	0	0	0

Table 5.3 Link parameters (continued) ($N.cm^2$)

Link	I_{xx}	I_{yy}	I_{zz}
1	148063.3	148063.3	19008.6
2	428540.4	1399176.4	1066616.3
3	239341.9	241583.0	12806.6
4	2771.6	1850.7	2830.4
5	161.6	262.3	161.6
6	331.2	331.2	8.9

The definitions of a, d in the table 5.2 are shown in the Fig.4.3, and the definition of α is shown in the Fig. 3.1.

In the table 5.1-5.3, $m, \overline{X}, \overline{Y}, \overline{Z}, I_{xx}, I_{yy}$ and I_{zz} are the mass, the coordinates of mass centre and the main moments of inertia of the corresponding links, respectively.

From the following formulae, we can compute the constant parameters in the procedure of the computation (see table 5.4-5.5).

$$M^i = \sum_{j=i}^{n} m_j$$

$$\overline{p}_i = \begin{pmatrix} a_i & d_i \sin\alpha_i & d_i \cos\alpha_i \end{pmatrix}^T = \begin{pmatrix} p_x & p_y & p_z \end{pmatrix}^T$$

$$d_{ii} = \overline{p}_i + \overline{G}_i = \begin{pmatrix} d_x & d_y & d_z \end{pmatrix}^T$$

$$\overline{u}_i = M^{i+1}\overline{p}_i + m_i \overline{d}_{ii} = \begin{pmatrix} u_x & u_y & u_z \end{pmatrix}^T$$

$$\underline{\underline{k}}_i = \underline{\underline{J}}_i - m_i(\hat{\underline{\underline{d}}}_{ii} \cdot \hat{\underline{\underline{d}}}_{ii}) - M^{i+1}(\hat{\underline{\underline{p}}}_i \cdot \hat{\underline{\underline{p}}}_i)$$

$$= \underline{\underline{J}}_i + m_i \begin{bmatrix} d_z^2 + d_y^2 & -d_x d_y & -d_x d_z \\ -d_x d_y & d_x^2 + d_z^2 & -d_y d_z \\ -d_x d_z & -d_y d_z & d_x^2 + d_y^2 \end{bmatrix}$$

$$+ M^{i+1} \begin{bmatrix} p_z^2 + p_y^2 & -p_x p_y & -p_x p_z \\ -p_x p_y & p_x^2 + p_z^2 & -p_y p_z \\ -p_x p_z & -p_y p_z & p_x^2 + p_y^2 \end{bmatrix}$$

Table 5.4 Constant parameters

Link	1	2	3	4	5	6
M^i	159.44	125.94	48.64	12.34	3.39	1.0
p_x	0	43.2	1.9	0	0	0
p_y	0	0	-12.5	-43.2	0	0
p_z	0	0	0	0	0	0
d_x	0	21.6	1.9	0	0	0
d_y	0	0	-12.5	-41.2	0	0
d_z	8.0	21.75	21.6	0	0	1.0
u_x	0	3770.93	92.42	0	0	0
u_y	0	0	-608.0	-515.19	0	0
u_z	268.0	1681.28	784.08	0	0	1.0

Table 5.5 Constant parameters (continued)

		1	2	3
k_1	1	17252.5	0	0
	2	0	17252.5	0
	3	0	0	1939.65
k_2	1	80296.34	0	-36315.54
	2	0	306179.8	0
	3	-36315.54	0	235677.4
k_3	1	48958.77	1155.2	-1489.75
	2	1155.2	41763.05	9801.0
	3	-1489.75	9801.0	9082.39
k_4	1	21801.46	0	0
	2	0	188.84	-0.001
	3	0	-0.001	21801.46
k_5	1	16.49	0	0
	2	0	26.77	0
	3	0	0	16.49
k_6	1	34.8	0	0
	2	0	34.8	0
	3	0	0	0.911

5.5.2 The computation of the direct dynamics

The computational procedure and the flowchart are shown in the subsection 5.3.1 and 5.3.4. The driving torques $\tau_i = \tau_i(q_1, \ldots \ldots, q_n, \dot{q}_1, \ldots \ldots, \dot{q}_n, t)$ are given as follows.

$$\tau_1 = 3(\frac{\pi}{3} - q_1) - 5\dot{q}_1,$$

$$\begin{aligned}\tau_2 = {} & (q_2 - \frac{\pi}{3}) + 3\dot{q}_2 + g\{[m_3 + m_4)q_3 + m_4 L_5]\sin q_2 \\ & + (m_5 L_6 + m_6 L_3)(\cos q_5 \sin q_2 + \cos q_4 \sin q_5 \cos q_2) \\ & + (m_5 + m_6)q_3 \sin q_2\},\end{aligned}$$

$$\begin{aligned}\tau_3 = {} & 0.3(q_3 - \frac{\pi}{3}) + 0.6\dot{q}_3 + g(m_5 L_6 + m_6 L_3)\cdot \\ & \cdot \sin q_2 \sin q_4 \sin q_5,\end{aligned}$$

$$\begin{aligned}\tau_4 = {} & 0.3(q_4 - \frac{\pi}{3}) + 0.6\dot{q}_4 + g(m_5 L_6 + m_6 L_3)\cdot \\ & \cdot \sin q_2 \sin q_4 \sin q_5,\end{aligned}$$

$$\begin{aligned}\tau_5 = {} & 0.3(q_5 - \frac{\pi}{3}) + 0.6\dot{q}_5 + g(m_5 L_6 + m_6 L_3)\cdot \\ & \cdot (\sin q_5 \sin q_2 + \sin q_2 \cos q_4 \cos q_5),\end{aligned}$$

$$\tau_6 = 0.25(q_6 - \frac{\pi}{3}) + 0.25\dot{q}_6.$$

In the computation, the step size HT is 10^{-4} second and the stop time M is 10 seconds. In the above formulae, the L_i depends on the coordinate of the mass centre and is the distance from the mass centre of link i to the origin of the coordinate system i. The initial conditions are as follows:

$$q_{i(0)} = \dot{q}_{i(0)} = 0 \qquad i = 1, 3, 4, 5, 6.$$

$$q_{2(0)} = \frac{\pi}{2}$$

The results of the computation are shown in the Fig. 5.4.

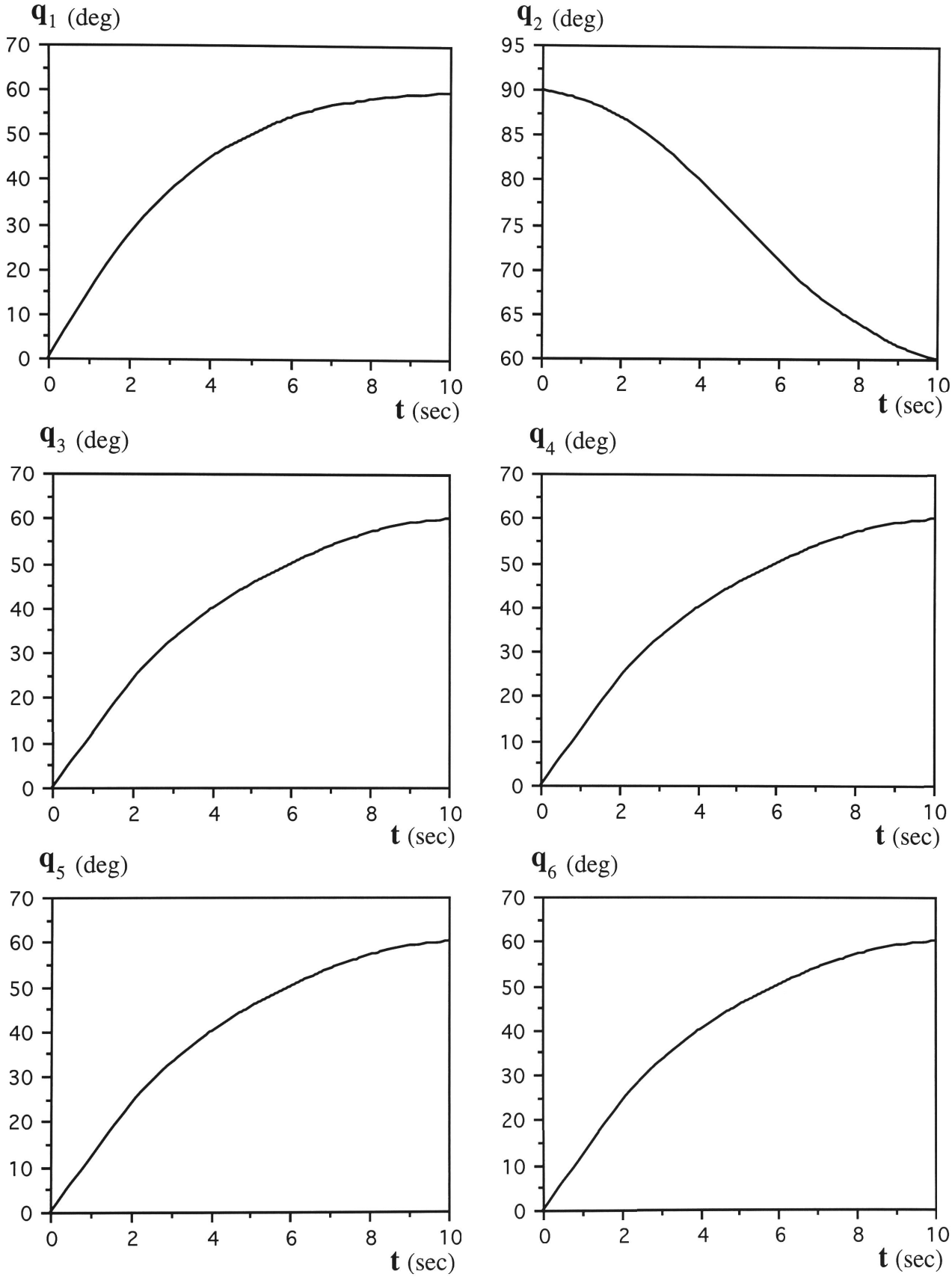

Fig. 5.4 The computation results of the direct dynamics

5.5.3 Computation of the inverse dynamics

The computational procedure and the flowchart are shown in the subsection 5.4.1 and 5.4.2. The motion laws $q_i = q_{i(t)}$ are given as follows.

$$q_i = \frac{\pi}{30}(t - \frac{5}{\pi}sin(\frac{\pi}{5}t)) \quad (rad) \quad i = 1, 4, 5, 6.$$

$$q_{2,3} = \frac{\pi}{2} - \frac{\pi}{60}(t - \frac{5}{\pi}sin(\frac{\pi}{5}t)) \quad (rad)$$

In the computation, the step size HT is 0.1 second and the stop time M is 10 seconds.

The results of computation are shown in the Fig. 5.5.

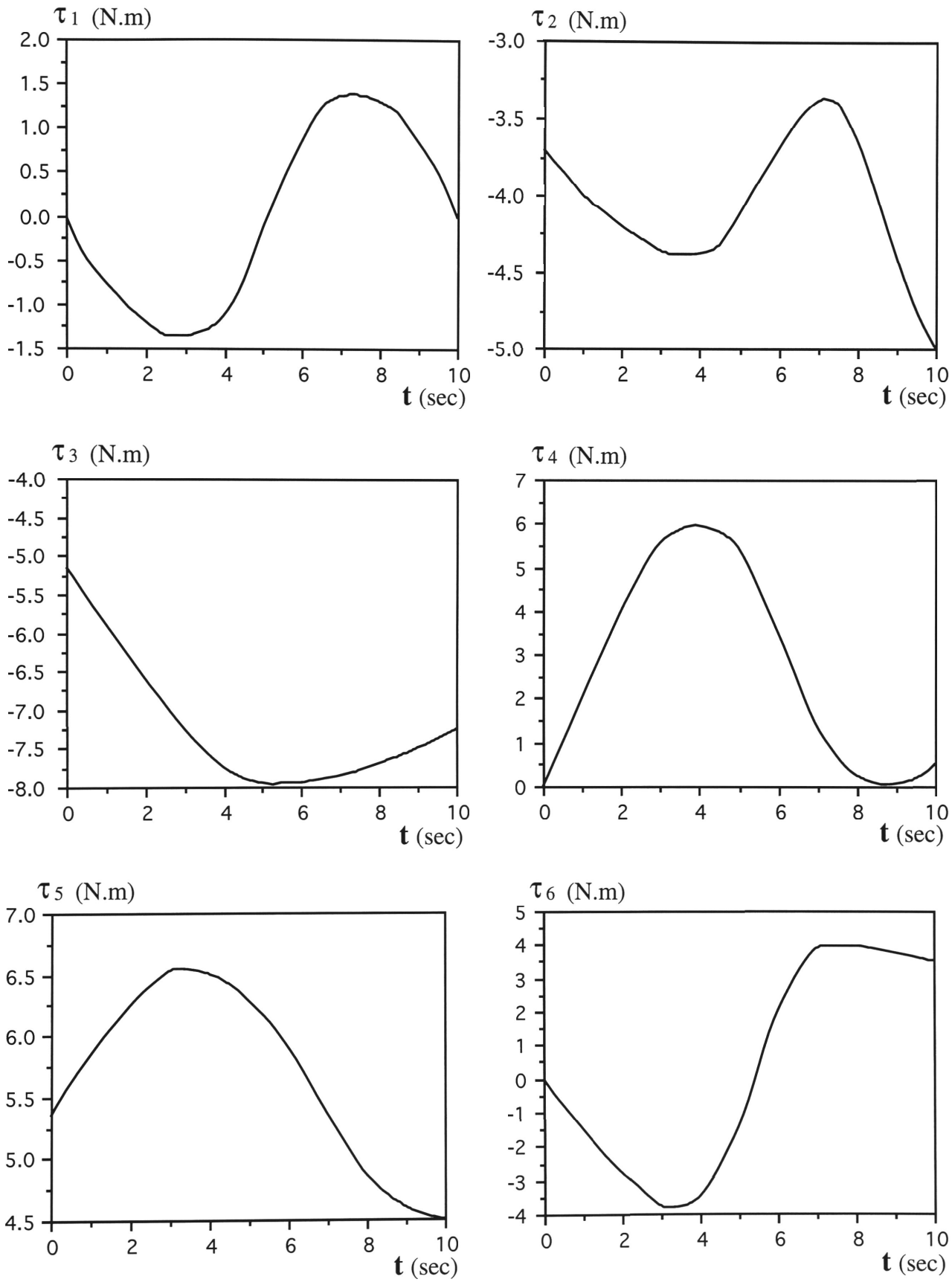

Fig. 5.5 The computation results of the inverse dynamics

CHAPTER 6

CONCLUSIONS

1. In this dissertation, a general form of dynamical equations for manipulators with revolute and/or prismatic joints is presented according to structural characteristics of the serial manipulators on the basis of multi-rigid-body system dynamics. The method presented can be used for any serial rigid body manipulator with revolute and/or prismatic joints.

2. An efficient recursive approach for computer generation of a manipulator model is presented. It suits the direct or inverse problems of dynamics. The model depends only on unit vectors and inertia tensors.

3. The equations are derived from D'Alembert's principle in which the ideal constrained forces are not considered. The equations obtained are only the explicit formulae of generalised coordinates q, $\dot{q}$ and $\ddot{q}$ and the forces or moments exerted by actuator at joints.

4. The equations can be formed and solved automatically by the computer. They don't need to be derived by hand as no partial differential equations appear. The equations can be built from known parameters such as body vectors and inertia tensors by recursive formulae. Elaborate calculation is avoided.

5. In the dynamical model, the augmented bodies and composite systems that suit serial manipulators are used. All constant parameters of the system can be summed up and computed in advance for known payload, which makes the computation simple and raises the computational efficiency.

6. Another characteristic of the model is that the kinematic vectors (linear velocity, acceleration, angular velocity and angular acceleration) are expressed by independent variables (generalised coordinates). Thus, the only variables of the equations are the generalised coordinates.

The model presented in this thesis does not consider a variation of payload. Depending on the payload, the main moments of inertia of the end-link of the manipulator will vary.

Though this presented method seems to be more efficient when compared with other modelling techniques with respect to the number of computation, simulations have to be performed to further prove it. Of course, experiments with actual robot also have to be performed to evaluate the method.

To sum up, the modelling technique presented in this dissertation results in a computational efficient model of a manipulator. It ought to be suitable for simulation purposes, to model the system during real time control and for an inverse model for actual techniques that require this. How to adjust the method presented in a computational efficient way is a topic for further research.

REFERENCES

1. A.H. Nayfeh and D.T. Mook, **Nonlinear Oscillation**, J. Wiley &Sons, NY, (1979).
2. J. Wittenburg, "Dynamics of multibody systems", **Proc. XVth IUTAM/ICTAM Congr**. (Toronto, 1980), pp.89-127.
3. K. Magnus (ed.), **Dynamics of Multibody Systems**, Proc. IUTAM Symposium, (Munchen,1977), Springer-Verlag, Berlin, (1978).
4. E.J. Haug (ed.), **Computer Aided Analysis and Optimization of Mechanical System Dynamics**, Proc. NATO Advanced Study Institute (Iowa City, 1983), Springer-Verlag, Berlin, (1984).
5. R. Schwertassek and R.E. Roberson, "A perspective on computer oriented multibody dynamic formalisms and their implementations", **Proc. IUTAM/IFToMM Symposium on Dynamics of Multibody Systems** (Sept. 1985, Udine, Italy), G. Bianchi, W. Schiehlen (eds.), Springer-Verlag, Berlin, (1986), pp.261-273.
6. R. Schwertassek, "The Roberson/Wittenburg formalism and the package MULTIBODY for computer simulations of multibody systems", European Space Agency, ESA-TT-557. (Nov 1979).
7. A. Jaschinski and W. Duffek, "Evaluation of bogie models with regard to dynamic curving performance of rail vehicles", **The Dynamics of Vehicles on Roads and on Railway Tracks**, Proc. 8th IAVSD Symposium (Cambridge, Mass.), Awets and Zeitlinger, Amsterdam, (1984).
8. K.N.Jr. Marman and F. Giannopoulos, "Recent advances in the analytical and computational aspects of modelling active and passive vehicle suspensions",

Computational Methods in Ground Transportation Vehicles, (M.M. Kamal, J.A. Wolf, eds.), ASME, AMD Vol.50, (1982), pp.75-115.

9. W. Kortum and A. Utzt, "Control law design and dynamic evaluations for a MAGLEV vehicle with combined lift and guidance", **Proc. 1983 American Control Conf**. San Francisco, (Jun 1983), Vol.1, pp.276-283.
10. D.A. Limbert, "Analysis and design of ferromagnetic suspensions for simultaneous lift and guidance of a tracked levitated vehicle", Sc.D. Dissertation, Department of Mechanical Engineering, MIT, Cambridge, Mass., (Dec 1976).
11. M. Vukobratovic and D. Stokic, **Control of Manipulation Robots**, Springer-Verlag, Berlin, (1982).
12. M.A. Chace, "Using DRAMand ADAMS programs to simulate machinery, vehicles", **Agricultural Engrg**. (Nov 1978), 18-19 and (Dec 1978), 16-18.
13. R.A. Wehage and E.J. Haug, "Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems", **Trans. ASME, J. of Mech.Design 104** (1982), pp.247-255.
14. R.L. Huston and C. Passerello, "On multi-rigid-body system dynamics", **Computers and Structures 10** (1979), pp.439-446.
15. P.N. Sheth and J.J. Uicker Jr., "A generalized symbolic notation for mechanisms", **J. of Engineering for Industry, Trans. AMSE** (Feb 1971), pp.102-112.
16. M.E. Kahn and B. Roth, "The near minimum time control of open loop articulated kinematic chains", **J. of Dynamic Systems, Measurement, and Control, Trans. ASME** (Sep 1971), pp.164-172.
17. M.H. Raibert, "Analytical equations vs table look-up for manipulation: A unifying concept", **Proc. IEEE Conference on Decision and Control,** New-Orleans, LA (1977) pp.576-579.

18. J.Y.S. Luh, M.W. Walker and R.P.C. Paul, "On line computational scheme for mechanical manipulators", **J. of Dynamic Systems, Measurement, and Control, Trans. ASME** Vol.102, (1980), pp.69-76.

19. Y. J. Fang and Z. Z. Wang, "A Computer-Oriented Recursive Method to Manipulator Dynamic Modelling", **Proc. of First International Applied Mechanical Systems Design Conference,** Nashville, Tennessee, U.S.A.. (Jun 1989).

20. R. Paul, **Robot manipulator: Mathematics, Programming, and control,** (MIT Press, Cambridge, MA, 1981).

21. C.P. Newman and J.J. Murray, "Computational robot dynamics: Foundations and applications", **J. of Robotics Systems** 2 (4) (1985) pp.425-452.

22. C.P. Newman and J.J. Murray, "The complete dynamic model and customized algorithms of the PUMA robot", **IEEE Trans. on Systems, Man and Cybernetics**, SMC-17 (4) (1987).

23. J.J. Murray and C.P. Newman, "Organizing customized robot dynamics algorithms for efficient numerical evaluation", **IEEE Trans. on Systems, Man and Cybernetics**, SMC-18 (1) (1988).

24. S. Megahed, "Force analysis of robot manipulators", **J. of Engineering Manufacture**, Part B, 203 (4) (1989) pp.217-232.

25. Y. J. Fang, A. Basu, X. D. Fang and Z. Z. Wang, "A Efficient Recursive Approach For Computer Generation of Manipulator Dynamic Model". **Mathematical and Computer modelling,** Vol. 20, No.9, (1994), pp. 89-96 .

26. Y. J. Fang, A. Basu and X. D. Fang, "A New Manipulator Dynamics Modelling Method", **Proc. of Third International Conference on Automation, Robotics and Computer Vision,** Singapore (Nov 1994).

27. W.M. Silver, "On the equivalence of Lagrangian and Newton-Euler dynamics for manipulators", **J. of Robotics Research**, 1, pp.118-128 (1982).

28. W.M. Armstrong, "Recursive solution to the equations of motion of an n-link manipulator", **Proc. the 5th World Congress, Theory of Machines, Mechanisms**, Vol. 2, pp.1343-1346, (Jul 1979).

29. M.W. Walker, "A Unified Approach to Manipulator Modelling", **IEEE CH 2152-7**, (1985).

30. J.M. Hollerbach, "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamic formulation complexity", **IEEE Trans. on Systems, Man and Cybernetics**, SMC-10, pp.730-736 (1980).

31. M.W. Walker and D.E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms", **J. of Dynamic Systems, Measurement, and Control, Trans. ASME** Vol. 104, pp.205-211, (Sep 1982).

32. J.J. Murray and C.P. Neuman, "ARM: An Algebraic Robot Dynamic Modelling Program", **Proc. 1st IEEE Conf. on Robotics**, Atlanta, (1984), pp.103-114.

33. E.J. Kreuzer and W.O. Schiehlen, "Computerized Generation of Symbolic Equations of Motions of Motion for Spacecraft", **J. Guidance and Control**, Vol. 8, (1985), pp.284-287.

34. J. Wittenburg and U. Wolz, "MESA VERDE: A Symbolic Program for Nonlinear Articulated Rigid Bodies Dynamics", **ASME Paper No. 85-DET-151**.

35. D.E. Rosenthal and M.A. Sherman, "High Performance Multibody Simulation Via Symbolic Equation Manipulation and Kane's Method", **J. Astronaut. Science**, Vol. 34, (1986), pp.223-239.

36. D.B. Schaechter and D.A. Levinson, "Interactive Computerized Symbolic Dynamics for the Dynamicists", **Proc. ACC**, Atlanta, (1988), pp.177-188.

37. P.Y. Cheng, C.I. Weng and C.K. Chen, "Symbolic Derivation of Dynamic Equations of Motion for Robot Manipulator Using Program Symbolic Method", **IEEE J. Robotics and Automation**, Vol.4, (1988), pp.599-609.

38. M. Vukobratovic and N.Kircanski. "A Method for Computer-Aided Construction of Analytical Models of Robotic manipulators", **IEEE CH2008-1**, (1984).

39. J. Wittenburg, **Dynamics of Systems of Rigid Bodies**, Stuttgart, B.G.Teubner, 1977.

40. J. Denavit and R.S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices", **ASME J. of Applied Mechanics**, (Jun 1955), pp.215-221.

41. J. Koplik and M.C. Leu, "Computer Generation of Robot Dynamics Equations and the Related Issues", **J. of Robotic Systems**, (3), (1986)

42. K.N. Reid, "Research Needs in Mechanical Systems", **Trans. ASME, J. of Mechanisms, Transmissions, and Automation in Design**, Vol.106, (Jun 1984).

43. C.A. Balafoutis, R.V. Patel, "Efficient Computation of Manipulator Inertia Matrices and the Direct Dynamics Problem", **IEEE Trans. on Systems, Man and Cybernetics**, Vol.19, n5, (Sep-Oct 1989).

44. A. Jain and G. Rodriguez, "Recursive Linearization of Manipulator Dynamics Models" **IEEE CH2930-6**, (1990).

45. I.B. Baharin, R.J. Green, "Computationally-effective Recursive Lagrangian Formulation of Manipulation Dynamics", **J. of Control**, Vol.54, n1, (Jul 1991).

46. J. Lieh, "Computer-aided Modelling and Control of Multibody Systems for Robotics Application", **IEEE CH2969-4**, (1991).

47. R.P. Paul, M. Rong and H. Zhang, "The Dynamics of the PUMA Manipulator", **Proc. American Control**, pp.491-459, (1983).

48. Liu Yan-zhu, **The Dynamics of Multi-Rigid-Body Systems**, Press of Shanghai Jiaotong University, P.R.China, (1985).

49. Zhou Qi-zhao, **Multi-Rigid-Body Dynamics**, Press of Peking University, P.R.China, (1985).

50. Wu Fei, "Study of Dynamics Equations of Robot Manipulators", **Proc. 6th National Mechanisms Conference of CMES**, P.R.China, (1988).

51. J.J. Uicker, Jr, "Dynamic Force Analysis Of Spatial Linkages", **Trans.ASME, J. of Applied Mechanics**, 34, pp.418-424, (1967).

52. M.E. Kahn, "**The Near Minimum-Time Control Of Open-Loop Articulated Kinematic Chains**", Stanford Artificial Intelligence Laboratory, AIM-106, (1969).

53 A.K. Bejczy, "**Robot Arm Dynamics And Control**", Technical Memorandum 33-669, Jet Propulsion Laboratory, Pasadena, California, (1974).

54. R.P. Paul, "**Robot Manipulators: Mathematics, Programming, and Control**", MIT Press, Cambridge, Mass, (1981).

55. D.E. Orin, R.B. McGhee, M. Vukobratovic, and G. Hartoch, "**Kinematic and Kinetic Analysis of Open-Chain Linkages Utilising Newton-Euler Methods**", Mathematical BioSciences, 43, pp.107-130, (1979).

56. M.H. Raibert, and B.K. Horn, "**Manipulator Control Using The Configuration Space Method**", Industrial Robot, 5, pp.69-73, (1978).

57. T.R. Kane, and D.A. Levinson, "**The Use of Kane's Dynamical Equations In Robotics**", International Journal of Robotics Research, 2, pp.3-21, (1983).

58. M. Vukobratovic, and V. Potkonjak, "**Two New Methods for Computer Forming Of Dynamic Equations of Active Mechanisms**", Journal of Mechanisms and Machine Theory, 14, (1979),

59. C. S. G. Lee, and B. H. Lee, "**Development of the Generalised D'Alembert Equations of Motion for Robot Manipulators**", IEEE, Trans. on Systems, Man and Cybernetics, 17, pp.311-325, (1987).

60. L. H. Lathrop, "Parallelism in Manipulator Dynamics", MIT Artificial Intelligence Laboratory, Cambridge, Mass, Technical Report No.754, (1983).

61. C. S. Lee, and P. R. Chang, "Efficient Parallel Algorithm for Robot Inverse Dynamics Computation", IEEE Trans. on Systems, Man and Cybernetics, 16, pp.532-542, (1986).

62. E. E. Binder, and J. H. Herzog, "Distributed Computer architecture and Fast Parallel Algorithms in Real-Time Robot Control", IEEE Trans. on Systems, Man and Cybernetics, 16, pp.534-549, (1986).

63. J. J. Murray, and C. P. Neuman, "Organising Customised Robot Dynamics algorithms for Efficient Numerical Evaluation", IEEE Trans. on Systems, Man and Cybernetics, 18, pp.115-125, (1988).

64. P. Y. Cheng, C. I. Weng, and C. K. Chen, "Symbolic Derivation Of Dynamic Equations of Motion for Robot Manipulators Using Piogram Symbolic Method", IEEE Journal of Robotics and Automation, 4, pp.599-609, (1988).


Allbook Bindery
21 Pyramid Road
West Ryde 2114
Phone: 807 4031